PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 17/60

A2

(11) International Publication Number: **WO 00/46723**

(43) International Publication Date: 10 August 2000 (10.08.00)

(21) International Application Number: PCT/US00/02933

(22) International Filing Date: 3 February 2000 (03.02.00)

(30) Priority Data:

60/118,493	3 February 1999 (03.02.99)	US
09/496,530	2 February 2000 (02.02.00)	US

(71) Applicant: ONESOFT CORPORATION [US/US]; Suite 250, 7010 Little River Turnpike, Annandale, VA 22003–9998 (US).

(72) Inventors: MACINTYRE, James, W.; 4613 Hillbrook Drive, Annandale, VA 22003 (US). KOBEISSI, Said; 5810 Wood Poppy Court, Burke, VA 22015 (US). PARKER, Eric; 245 Summer Street, Somerville, MA 02143 (US). MONTAN, Vasile; 6008 Lincolna Road, Alexandria, VA 22312 (US). BAILEY, Robert, L.; 8613 Running Fox Court, Fairfax Station, VA 22039 (US).

(74) Agent: MIRABITO, A., Jason; Mintz Levin Cohn Ferris Glovsky and Popeo PC, One Financial Center, Boston, MA 02111 (US).

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published

Without international search report and to be republished upon receipt of that report.

(54) Title: MODULAR SYSTEM AND METHOD FOR PROCESSING TRANSACTIONS

(57) Abstract

A modular system for developing and processing network based transactions is made up of a four dimensional architecture or framework which can include four types of functional components or objects. These components include data management objects which access and move data within the system as part of the transaction, functional objects that can be used to transform or process the data, presentation objects which provide an interface to the customer or client in order to facilitate the transfer of information and control objects which determine when and how the data objects, functional objects and presentation objects should be applied to the data as part of the transaction. The system utilizes a standardized, extensible data structure for transferring the data between components or objects which allows the objects to be used as interchangeable building blocks of a comprehensive and flexible system architecture.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

MODULAR SYSTEM AND METHOD FOR PROCESSING TRANSACTIONS

COPYRIGHT NOTICE

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Application No. 60/118493, filed February 3, 1999, which application and its appendices are hereby incorporated by reference in their entirety.

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH

Not Applicable

REFERENCE TO MICROFICHE APPENDIX

Not Applicable

BACKGROUND OF THE INVENTION

This invention relates to a computer system for processing transactions over a network and, more particularly, to a system which utilizes modular, data independent, components to perform transaction processing functions.

Conventional transaction processing systems are typically constructed by developing a custom system, based upon customer requirements, around a developer's core technology. The resulting system essentially implements a customer's business model in a transaction processing system. For each business model, a new system is developed around the same core technology.

In addition, a system developed for the sale of hardgoods, such as clothing or food is likely going to be very different from a system that provides access to a chat system or sells music or stock or options.

One of the disadvantages of this methodology is that, depending upon each customer's requirements, a custom system must be developed for each business model and changes to this custom system will require an investment in a new and substantial development effort. For example, a system may be developed to receive an order for a product, then ship the order and send out an invoice. Substantial effort would be required to add encryption to the transaction processing to enhance security, or give the system the ability to invoice an order and delay shipment until payment is received (for customers with poor payment history) or ship and invoice substitute products when an ordered product is discontinued or out of stock.

Accordingly, it is an object of this invention to provide an improved system for processing transactions.

It is another object of this invention to provide an improved system for processing transactions which is modular, data independent and can be easily adapted to accommodate varied customer business models and well as to be easily modified to accommodate changes in business model requirements.

SUMMARY OF THE INVENTION

The present invention is directed to methods of and systems for processing transactions that incorporate a four dimensional system architecture. This architecture divides the system into four basic types of modular components or program objects. These component types include: data management components or objects which can be

used to manipulate data within the system as part of the transaction; functional components or objects which can be used to transform or process the data; presentation components or objects which can be used to provide an interface to the customer or client in order to facilitate the transfer of information; and control components or objects which determine when and how the data objects, functional objects and presentation objects should be applied to implement and process the transaction. In accordance with a preferred embodiment of the present invention, the system utilizes a standardized, extensible data structure or interface for controlling the process flow as well as transferring the data between objects.

Alternatively, the present invention is directed to methods and systems for processing transactions that incorporate a four dimensional command architecture. This command architecture includes a system of modular commands that provide the fundamental building blocks used to create a transaction processing system. In accordance with the present invention, this architecture divides the system commands into four basic types. These command types include: data management commands which can be used to manipulate data within the system as part of the transaction; service commands which can be used to transform or process the data; presentation commands which can be used to provide an interface to the customer or client in order to facilitate the transfer of information; and control commands which determine when and how the data objects, functional objects and presentation objects should be applied to implement and process the transaction.

In accordance with the present invention, the system processes a transaction that is defined by a business or transactional model. The control component or object uses the business model to control the order and manner in which the data objects, the

functional or service objects and the presentation objects are used to process the transaction. The control component interacts with the data objects, the functional objects and the presentation objects, by exchanging structured data records with the predefined data, functional and presentation objects, in a predefined manner according to the business or transaction model. This can be accomplished by invoking data objects, functional objects and presentation objects in a manner defined by the business or transactional model. Alternatively, this can be accomplished by invoking a set of commands in a predefined manner defined by the business or transactional model. The business or transaction model can be made up of one or more business or transactional actions which make up a session during which the system interacts with a customer. Each business or transactional action can involve invoking one or more of the system objects which can process input from the customer and generate a response which is transferred to the customer.

A higher level monitoring system utilizing additional functional and data objects can be integrated into the system to collect and report data relating to individual transactions and groups of transactions including providing forecasts and identifying trends, reporting business and financial information based upon several transactions.

In accordance with one embodiment of the present invention, the system can be configured to operate in a distributed processing environment where multiple servers, configured in an array, can be used to execute each instance of a component or program object of the system. The system can further include load balancing capability to distribute both the front-end processing load (servicing incoming requests) and back-side (application services) processing load evenly and optimally over all the available servers in the system. The system dynamically balances the load according to the capacity of

each individual server in the system, thus a server with twice the capacity will receive twice the load.

According to another embodiment of the invention, a method is provided for constructing a transaction processing system. The method includes the step of defining a business model for the system. The method also includes the step of using a controller to control system software components (elements or objects) as a function of the business model to perform a series of business actions. The series of business actions make up a transaction. Each of the business actions includes at least one function call. A function call includes the steps of invoking a software component, passing data to the software component, and executing a software function as a function of the data. In one embodiment, the function call passes to the software component data in the form of structured data records utilizing an extensible markup language or a hypertext markup language such as may be used in a web page.

In another embodiment of the invention, the system for processing a transaction includes a computer processing system and associated memory, a plurality of independent program objects operatively coupled to the computer processing system, wherein each object is selected from a set of different program object types. The group of program object types consist of application object types adapted for providing data management services, service object types adapted for providing data transformation and monitoring services, presentation object types adapted for providing data presentation services to a participant in the transaction, and control object types adapted for controlling each of the other independent program objects for processing the transaction. Each of the independent program objects is adapted for communicating with another independent program object according to a common extensible interface or data

structure. In addition, any program object can be added or removed from the system without affecting the operation of the other program objects.

The business model can be defined as a set of business actions and each business action can be defined by set of object invocations or instances, which are used by a control component, or control commands to execute various functions provided by the other system components. The business action can be provided in the form of one or more markup language pages, each including one or more calls to any of the available system components or system commands. A business action can be initiated by a customer requesting access to a specific web document residing on a web server. In response to the request for the specified web document, the web server submits the corresponding markup language page (which defines the appropriate business action) to the controller. The controller parses the calls or commands to the system components or function in the order specified to complete the business action. When the business action is complete, a presentation component or function transmits the appropriate web page to the web server to be delivered to the customer.

Each of the system components can be preconfigured to support auditing and monitoring functions. Thus, the system can track and log every system component accessed or every system function performed. Specifically, the system can monitor every piece of content viewed as well as every product or service item viewed by a given customer. In addition, the invention further contemplates having the system developer assign or associate a customer category or rating with every piece of content and every product and service item available, such that the system can create a profile for each

customer as a function of each item of content viewed by a customer to allow the system to target products and services that are more suited to the customer's needs. Thus, every time a customer requests a specific web page, the system can update the customer's profile as a function of the category or rating for the content of the page requested.

In addition, the system can further include predefined auditing and monitoring functions that relate to the financial status of the transactional processing system. The system can monitor the goods and services sold and report, in real time, on the financial health of system on a business model by business model basis or for the system overall. Specifically, the system can report the profit and loss for a given business model, the total cost of ownership of the system and the return on investment of the system. This can be accomplished by enabling the system components to report transactional data to specialized financial modeling components which use the transaction data and other data (such as wholesale costs, margins, channel costs, return costs and execution costs) to continuously report the financial health of a business model or the entire system at any point in time.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects of this invention, the various features thereof, as well as the invention itself, may be more fully understood from the following description, when read together with the accompanying drawings in which:

FIGURE 1 is a diagrammatic view of the architecture of an e-commerce system according to one embodiment of the present invention;

FIGURE 2 is a diagrammatic view of a business model according to one embodiment of the present invention;

FIGURE 3 is a flowchart according to one embodiment of the invention for constructing the e-commerce system of FIGURE 1;

FIGURE 4 is a diagram of functions that make up a business action according to one embodiment of the present invention;

FIGURE 5 is a sample data item that holds information used by a service component;

FIGURE 6 is a sample data item that holds a user request information to be used by a service component;

FIGURE 7 is a sample data item that holds an audit trail of all actions that were performed during a session;

FIGURE 8 is a diagrammatic representation of a system for processing transactions according to one embodiment of the present invention;

FIGURE 9 is an example of an active server page (ASP) file which handles the request to view a targeted catalog; and

FIGURE 10 is an example of a data item which is managed by a Product Application Component type (ACT).

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention is directed to methods of, and systems for, processing transactions. In order to facilitate a further understanding of the invention, a system for conducting Internet or electronic commerce ("e-commerce") embodying the methods and systems of the present invention is described herein.

FIGURE 1 shows a modular system 100 according to the present invention which includes a plurality of components or program objects. The modular system 100 organizes the system program components or objects into one of four basic dimensions herein referred to as system component types. These component types can include: application component types (ACTs) 130 which include program components or objects 132 that provide persistent data access and management functions; service component types (SCTs) 120 which include program components or objects 122 that can be used to provide services, for example, which manipulate, analyze and/or transform data; control component types or business model component types (BCTs) 112 which include program components or objects 114 that can be used to control the logical flow of a transaction; and determine when and how the other program components or objects are used, and presentation component types (PCTs) 140 which include program components or objects 142 that define how data is presented to entities external to the system such as a customer. The SCTs can include a financial modeling component 122A and a tracking and profiling component 122B that are described in further detail below.

The system can further include connector component types (CCTs) 160 and workstation component types (WCTs) 170. CCTs 160 include connector program components or objects 162 that interface with legacy systems 164. CCTs also interface with systems that can provide access to external services 166. CCTs that provide access to legacy data can be considered ACTs for purposes of this disclosure. CCTs convert data from those legacy systems into the common data representation language used by the present invention. CCTs that provide access to external services can be considered

SCTs for purposes of this disclosure. WCTs are BCTs that use available system data (provided by ACTs as may be modified by SCTs) in order to perform administrative functions on the system.

The BCTs 112 are part of the system control function or controller 110 which implements an e-commerce system according to one or more business models 200. A business model 200 includes one or more business actions 210. The business model 200 informs the system controller 110 which data objects are to be used, which functions or services are to be performed (and in what order) and which presentation formats to use in a given business action. In addition, the functional objects can be independent of the data elements and interchangeable with each other. In accordance with one embodiment of the invention, the data objects are implemented in the form of application component types and the functional elements are implemented in the form of service component types. Preferably, SCTs can apply their services (e.g., cataloging, sorting, and searching) polymorphically to all types of data, i.e., to all ACTs.

Preferably, each of the ACTs 130, SCTs 120, BCTs 112 and PCTs 140 conform to a predefined program component or object protocol which defines the structure of the data elements and interfaces. In the illustrative embodiment, all the components of a given type are completely interchangeable with each other and can interact with components of any other type. In addition, each of the ACTs 130, SCTs 120, BCTs 112 and PCTs 140 can utilize a common data structure and data interchange format 150 to facilitate the transfer of information between components.

Typically, a customer accesses a vendor web site residing on a web server 180 using a browser on the customer's system 182 and software executed on the vendor's server system 182. From the customer's perspective, a typical vendor website is a

network of related documents. The customer can navigate from one document to the next by clicking on document links. Each time the customer clicks on a document link he is submitting a request to a web server 180 to retrieve a document from the web server 180. The customer is normally unaware of which web server 180 he is submitting the request to, and the web server 180 is normally not concerned with how each request is related to previous requests.

The e-commerce system 100 according to the present invention extends this idea by generating documents dynamically as they are requested. Through the use of HTML forms and "Cookies," the web server 180 collects customer information from the customer system 182 and passes it to the e-commerce system 100 along with the request. In accordance with the present invention, the web server's responds to this request differently than it would respond to a request for a simple web page. This request can invoke a complex set of objects or functions that generate predefined output to be displayed on the customer's browser. The e-commerce system 100 can use the received customer information (such as from HTML forms and "Cookies" that are received from the customer system 180 via the browser) to maintain state information which allows the system 100 to group together other requests into a series of related requests known as a session or transaction. The related requests that make up a session are typically made by the same browser and occur close enough together to be assumed to be made by the same customer. The system 100 can further require a customer to authenticate himself with a username and password or by presenting a certificate in order to be able to tie together a group of requests which are made by the same customer as part of a session and ultimately, a transaction. The system 100 can also use Cookies to track a customer in a session.

In accordance with the invention, an e-commerce system 100 incorporates one or more business models 200. As shown in FIGURE 2, each business model 200 is made up of a set of business actions 210 that the system 100 permits a customer to initiate. In one embodiment, the business actions 210 are performed by passing data in the form of at least one web page 220 between component types. In the illustrated embodiment, the business models are formatted according to Microsoft Active Server Page (ASP) format as defined by Microsoft Corporation, Redmond, Washington. However, the invention contemplates the use of any of a variety of common representation languages, including, Perl, COM objects written in C++, a variety of standardized markup languages and extensible formats, such as eXtensible Markup Language (XML), HTML, or SGML.

The resulting e-commerce system 100 can guide the customer through the steps of a transaction in a logical manner by providing links to appropriate 'next' actions as a result of successfully performing a prior action. However, the customer may bookmark a URL (a request) or may type a requested URL directly, so the e-commerce application cannot rely solely on the customer making the requests in a logical or expected order. A customer may request that a business action be performed at any time and it is the responsibility of the controller 110 to determine if it is appropriate to let the customer initiate the action.

Each time a customer submits a request, the server 180 initiates the performance of a business action, the controller 110 can: 1) determine whether the customer is permitted to perform this action; 2) determine whether the customer has supplied all the

necessary inputs to perform the action; 3) determine whether the customer has been given access to all data required to perform the action; and 4) determine whether the customer has successfully completed all prerequisite actions.

If all of these conditions are satisfied, then the controller 110 will permit the action to be performed. Otherwise the controller 110 must choose a more suitable action to perform. A more suitable action may be to request additional input, return an error message and ask the customer to choose an alternative action, attempt to process prerequisite actions, or any other action that the controller may determine is appropriate.

In accordance with one embodiment of the invention, the system 100 can include a more sophisticated controller 110 which can keep track of requested actions which were denied because of missing inputs or incomplete prerequisite actions so that it can perform these actions once the missing input is supplied or the prerequisite action is performed. For example, if a customer who must log in before viewing product information makes a view request before logging in, the controller 110 could request that the customer log in but save the prior request to view product information which can be performed after the customer has successfully logged in.

All of the business actions 210 available to customers of the system 100 as well as restrictions on what data the actions 210 require, what access controls are placed on the actions, etc. make up a Business Model 200. Using this information, the controller 110 can operate by dispatching or invoking system components or objects to implement the business action 210. In accordance with the invention, the system components can be designed to operate independently of each other and the business action in order to provide a separation between the control of the system and the functions that the system

can perform. Services (for example, a "search" service) can be implemented without regard to the business data structure or logic that is associated with any given data object (for example, a "product catalog" or "chat room dialog"). New functions to be added and existing ones can be removed or replaced without affecting the performance of the rest of the system. The Business Model 200 itself to be replaced or modified as necessary to change the entire behavior of the system.

As shown in FIGURE 3, a method for constructing a system for conducting electronic commerce in accordance with the invention can include the following steps: A) defining a business or transactional model or models for the system, 300; B) utilizing each of the model(s) to select system components, i.e., application component types, service component types, presentation component types, and business model component types, 310; and C) assembling 320 the system components with core system components on a scaleable system platform (hardware and software), 320.

In accordance with present invention, the Business Model 200 can be created easily by a person having limited programming skills and can be easily swapped in and out of an otherwise complete system. The system architecture which uses a business model and a control component in accordance with the invention permits a website developer to put together a site by simply specifying the actions that a customer can perform. Preferably, there is limited interdependence between the control system and the functional components since the code that checks for prerequisite actions or data is not mixed with the code that performs the business action.

A Business Model Utility can be provided to assist a web site developer in creating an e-commerce system 100 just by pulling together all of the reusable

application, service, presentation and control components provided by the system into a logical flow of control. This can be accomplished by providing a "picklist" of functions that a site developer can select and order as desired. Alternatively, a "drag and drop" graphical interface can be provided to allow a site developer to create Business Models by arranging symbols representative of components in a manner similar to the way a person would arrange the symbols of a flow chart.

The system 100 according to the invention performs each Business Action 400 that the customer requests by invoking a series of services (service functions or methods) as shown in the example of FIGURE 4. The illustrated business action begins with an initial web server request 410. The request 410 can be a request for a web page that is transmitted by the customer's browser. The actual page requested can be an ASP page that is used by Microsoft Internet Information Server (IIS) to invoke one or more ACTs, SCTs, or PCTs, to render a response in the form of an HTML page that is returned to the customer. The components can be implemented in the form of objects that conform to the Microsoft Component Object Model (COM) specification and executed on the Microsoft Transaction Server (MTS) platform. The web server request causes the business action 400 to call the component OneSession 430 and to invoke the function GetRequest of the component OneSession 430. The component OneSession 430 then passes data in the form of an XML page 412 to another component OneCatalog 432 in accordance with the business action 400. The component OneCatalog 432 invokes the GetItem function to obtain the catalog data for one or more product items (to be viewed or purchased) and appends the data to the XML page 416. Information concerning the product item or items can be received from the request as well as obtained when the component OneSession 430 restored the session data in the XML page as part of the

GetRequest function. The component OneTarget 434 invokes the Target function to add any targeting information to the XML page 418. The component OneAudit 436 invokes the log function to log the state of the transaction from the XML page 420. The component OneStore 438 uses the XML page 420 and style sheet such as eXtensible Style Sheet Language (XSL) formatted style sheet to render an HTML page that is transmitted to the customer containing content about the product items requested by the customer as well as any targeting information (e.g. product specials or related products). This process of calling components and passing data in the form of XML pages continues until the business action is complete.

A controller 110 uses a business model to provide the logic which controls the flow through web requests to a website and several controller/business models can be executed concurrently. In the illustrative embodiment, the system 100 can execute a separate business model for each customer accessing the website. Preferably, the initial input to the web server is a customer request which can spawn a separate business model instance to service each customer. The controller can be implemented as: a library of ASP functions (a business action can be an ASP page or a set of ASP pages), an ASP function which calls a COM or DCOM object, a Java serverlet, a CGI application, an ISAPI application, or an ISAPI filter. An e-commerce web site application can contain one or more business models or be limited to only one business model.

Preferably, all service components regardless of the functionality that they implement take the same form. Service components can act as data filters that take (XML) structured data as input, transform or process the data, and produce (XML) structured data as an output. In this way, services or functions can be chained together so that the output of one service or filter may be the input of the next service or filter. A

business action can be built by sequencing service components together with data manipulation functions in such a way that the action starts with the customer's request and ends with the HTML output by a presentation component that will be displayed on the customer's browser.

The Business Model specifies what services are invoked for each business action and what order the services or functions need to be executed. The controller makes sure that the services are executed in order and that the necessary data is passed from one service to the next as needed according to the business model. There is a possibility for each of the services to fail for some reason or other so the business model must support error handling which provides an alternative set of services to use in the case of failure.

In accordance with the invention, the data transferred between components is structured using a common data representation format language for all component types. Preferably, all the ACTs, SCTs, and PCTs conform to this common data representation model and use this common data representation language so that any component or object can interact with any other component or object to utilize any data which is passed to it. In the illustrative embodiment, the e-commerce system 100 uses XML as the common data representation language for all component types and all components are designed to communicate data structured according to a predefined XML schema.

In accordance with the invention, the service components can access other service components and application components in addition to the data that the services receive from the controller. Accessing additional data from other application components allows a service component to merge together data from multiple sources such as other services, application components and external systems. Accessing additional service components permits a single service component to implement complex service functions by

constructing a "pipeline" or sequence of functions which permit the complex Business Models to be easily created from a series of basic components. For example, the component OneAudit 436 can be a service component that parses the XML page 418 in order to: 1) pass session state information to a OneSession ACT which stores the session data; 2) pass the catalog data to a OneProfile SCT which creates customer profile information and updates the customer profile stored in the OneProfile ACT; and 3) pass audit information to a OneAudit ACT which tracks the status of the customer transaction.

In addition, as shown in FIGURE 1, additional component types such as CCTs 160 which interface the e-commerce system 100 with external services 166 and legacy systems and databases 164 can be included in accordance with the invention. The connector components 162 receive data from an external source, such as a legacy system 164, in the external source's native format and convert the data to the common data representation language used by the system 100. Such data conversion allows the e-commerce system 100 to use the data as if the data is native to the system 100. The connector component types 160 can similarly access the services of an external system 166 in a way that conforms to the data and interface requirements of a service component type 120 allowing the e-commerce system 100 to use these external services 166 as if they are native to the system.

The types of services that can be provided within this framework are extensive and can include for example: Fill out an empty data structure with data from the various application components (data repositories); Remove data that the customer does not have access to; Create a log as to what transformations have taken place; Update customer targeting information based on the customer's request for a specific web page or more information; Add in new information about products that are related to selected

ones or targeted to the current customer; Automatically discount the price of a product based on previous purchases; Sort data based on a customer's preferences or targeting profile; Determine the appropriate view for the resulting data and the customer's browser type and convert the data to an HTML format.

Service Components

In the illustrative embodiment, the service components essentially fit into one of several different categories: Request Processing Services, Data Transformation Services, and Monitoring Services. However, as a person having ordinary skill in the art will appreciate, the nature and types of services are extensible and additional categories can be added.

Request Processing Services.

Preferably, every business action has to start with a request for a web page received from the customer. The request (for example, HTML form data) can be converted into an XML data structure to begin the business action or data received as part of the request (form data, cookie data, etc.) can be used to identify the customer and create an XML representation of a user session. Request Processing Services are unique in that they are the only services that do not require XML as an input. Request Processing Services can take data directly from the variables available in the web server and can create XML structure which includes all or some of this data. The Server variables come from a variety of places, for example the URL which contains the path and name of the HTML, ASP, or other page that is being requested. It can also contain any number of request variables. The server variables can come from the HTML form values which can be passed in either as a GET request function or a POST request

function. In a GET request, the values are passed in on the URL. In a POST request, the values are passed in on the standard input. All of these values are available to the web server and request being processed according to the business model. The server values can also come from cookies which get passed from the server to customer's computer and allow the server to track the customer throughout the system. In addition, the server values can come from system variables which are accessible from the web server through an operating system, a gateway interface or a hardware platform. For example, the OneAccess service controls the level of access each user has within the system 100.

Data Transformation Services

Data Transformation Services modify the data that is received by adding additional data, removing data, or changing data values. These services are typically only interested in a part or subset of the data that is passed in, although all the data in the XML page can be modified. They make their transformations to one or more predetermined data elements or fields, if they are present at all, and pass the other data through unmodified.

Monitoring Services

Monitoring Services typically do not modify the data. They pass it though, but extract predefined portions of the data to be saved in audit or log stores (ACTs), update the system or session state, or store in a persistent storage. The extracted data can be used for system performance monitoring and reporting.

Service components provide the functionality of the system 100. In accordance with the invention, each service component can contain its own data and may not even

require data at all. The system architecture provides for all service components to be applicable across all ACTs, but many service components can be applied only to specified types of data.

Preferably, an SCT utilizes one or more ACTs and exposes none of its own data, but can generate some non-ACT data to be displayed. SCTs can provide helping functions so the other components do not need to manipulate XML and SCTs may combine data from multiple ACTs. In one embodiment, each SCT function applies across all ACTs. Alternatively, the system may be designed whereby not all SCT functions apply across all ACTs. Preferably, there can be a class of ACTs which are 'salable' which can be used with any pure commerce function. In addition, a service component can have more than one function, service or method and a service component can use other service components.

The installation of a new SCT or a group of new SCTs can include an SCT installation package can be made up of the following: Documentation; a list of basic ACTs; a methods library, for example: OneItemSCT.dll (SCT interface with 1 or more methods defined); Registry Settings; Sample ASP file for each of the defined functions; and Default XSL views to edit and view any data generated/compiled by a services.

Many different types of data can be used in an e-commerce system, for example: customer information in an LDAP database, product information in a SQL Server database, Publications from a file system, news and securities information from a push data server, streaming video/audio from a multimedia database, and advisor information from an expert system.

In order to simplify how system components or objects access and store this data, all data is converted into a common data format, such as XML. This enables the system

components, objects and instances to be completely reusable even when the data they process comes from a vastly different data source and represents vastly different real world objects. For example, in one e-commerce system, the same data structure and components that can be used in one system to sell clothing can be used in another system to sell paint. The system can use a connector component to interface with a new data source and convert the data from the native representation to the common data format. This enables the e-commerce system 100 to use legacy data or access external services by using an interface component.

Preferably, each single data repository is encapsulated in an Application Component Type, such as an application object that can provide data methods which can include creating, deleting, retrieving, updating, and searching for a particular type of data, e.g., product data. These services are different than the services provided by SCTs in that they do not process data. They either accept new or updated data for storage and return nothing, or accept identifiers or query information and return data from their repository. In one embodiment, the data that is transmitted to or received from an ACT is in XML format.

All data in the system is accessed through an Application Component that allows the system to treat that data as if it resides in an XML data repository. The Application Component provides a core set of data services that may be used by other services to save and retrieve data.

All application components manage their own data and do not rely on the existence of other application components or services. An Application Component can sit on top of many different types of information management system depending on any

number of factors. Examples of Application Component Types (ACTs) are provided in the table below:

Information Type	Information Management System
Hard Good Product	Relational Database
Login Information	Microsoft LDAP Implementation
Publications	File System
Stock/Commodity Quotes	Push Data Server
Streaming Video/Audio	Multimedia Database
Personal Advisor	Expert System
Pattern Recognition	Neural Network
Complex, Configurable Products	Object-Oriented Database

Preferably, all application components can include the ability to process their data in an XML format, however, it is not necessary that the ACT support an extensible schema. Preferably, all ACTs are provide basic functionality such as saving and retrieving a set of basic data attributes and elements. The ACT may support additional elements as needed. In one embodiment, the set of basic data attributes and elements is as follows:

Basic attributes

ID	A unique string that identifies this data item within the type.
TYPE	String identifying the type of data.
VERSION	String identifying the version of the ACT that this data item was created under.
SITE	Identifier of site that this data item belongs to.

Basic Elements

NAME	Short descriptive name of the data item.
DESCRIPTION	Longer descriptive name of the data item.
PRICE	Base price of the good/service
CREATOR	Identifier of user who created this data item.

CREATION_DATE	Date and time that the data item was created.
UPDATED_DATE	Date and time that the data item was last updated.
UPDATED_BY	Identifier of the user who last updated the data item.
STATUS	Current status of the data item.
VISIBLE	Indicates if the data has a visual component or not.
IMAGE_URL	URL of an image to associate with this data item.

Figure 10 shows an example of a data item which is managed by the OneProduct ACT.

ACT Data Services or Functions

Preferably, all Application Components support a common ACT Interface which provides predefined services or functions (methods), for example:

Create - the Create service creates a new data item and initializes it with the values specified in the input and saves it to persistent storage. It returns an item identifier to the data that it created. Regardless of the identifier that is passed in, Create will generate a new, unique identifier and assign it to the new data item. Create in one embodiment takes the following form:

Create([in] BSTR xmlOneItem, [out,retval] IDTYPE* xmlOneItemId)

Destroy - the Destroy service removes the specified data item from persistent storage. The only pertinent data in the input is the identifier of the item to remove. Destroy can take the following form: Destroy([in] IDTYPE xmlOneItemId)

GetItem - the GetItem service retrieves all or pieces of the specified data item from persistent storage and returns it to the calling component. The input data may be an item identifier or it may be an empty shell containing elements that are expected to be retrieved. GetItem uses the identifier to determine which data item to retrieve and uses

the input data as a guide on which elements to retrieve and send back. GetItem can take the following form:

GetItem([in] BSTR xmlOnItemIn, [out,retval] BSTR* xmlOneItemOut)

SetItem - the SetItem service saves the specified data item to persistent storage overwriting any existing data for this data item. If the entire data item is not specified then only those elements which are specified will be overwritten and the remaining element values will remain the same. SetItem can take the following form: SetItem([in] BSTR xmlOneItem)

GetCollection - the GetCollection service retrieves all data items that match the data item on the input. The result is a OneCollection type item which contains all of the data items which match the specified one. The collection may be empty if no matches were found. GetCollection can take the following form:

GetCollection([in] BSTR xmlOnItemIn, [out,retval] BSTR* xmlOneItemOut)

Each application component can include an ACT installation package including the following: 1) Documentation; 2) Library: OneItemACT.dll (ACT interface with Create, Destroy, GetItem, SetItem, GetCollection methods); 3) Registry Settings; 4) Database (scripts to create the database and populate if needed); 5) Sample ASP files to create, delete, edit, view and query an item; and 6) Default XSL views to edit, list and view an item.

Application Components allow other components to save and retrieve in formation in persistent storage and transfer data to and from other components in the form of XML data. The ACTs implement the data spectrum of the system. An ACT can provide methods, services or functions, other than the core set identified above, as may required to manipulate the data. An ACT can contain any data in the system including

data that represents a salable item. The four dimensional independent component architecture that uses ACTs and a common data format allows the Services (SCTs) provided to be polymorphic. Thus, for example, a sort service component can sort any data type that any ACT can provide in the common data format, such as, XML. This benefit comes from that fact that all SCTs are designed to interface with any and all ACT data. Preferably, an ACT provides no functionality beyond a core of set data management functions: Create, Destroy, GetItem, SetItem, Search. Preferably, an ACT can have no dependencies on other system components. In addition, a utility for creating ACTs can be provided.

In the illustrative embodiment of the invention, exemplary components can include OneAudit, OneCatalog, OneChat, OneEvent, OneInbox, OneInquiry, OneLink, OneNote, OneOrder, OneProduct, and OneSession. The following is a brief description of each of the functions.

The OneAudit service component monitors and logs order, payment, and other customer transactions.

The OneCatalog service component is used to manage a collection of product or similar items in a organized hierarchy. This collection of product or similar items may represent a page in an on-line catalog, an aisle in an online storefront, or any other hierarchy of items. An individual catalog may contain items of varying application component types - a single catalog can include hard good offerings, discussion groups and chats, or any combination of available application component types that are all related to a common theme. The system 100 can also include a OneCatalog application component that maintains the data required by the OneCatalog service component.

The OneCatalog service component can include a plurality of functions or methods. The functions can include, for example, ADDItemCatalog, CreateCatalog, DeleteCatalog, Destroy, GetCollection, GetItem, ModifyCatalog, RemoveItemFromCatalog, RemoveCatalog, and SetItem. Example of various OneCatalog functions or methods are described below.

The AddItemToCatalog function adds the specified product item to the specified catalog section.

The CreateCatalog function creates an empty catalog within the specified parent catalog and returns its identifier.

The DeleteCatalog function removes the specified catalog from the system.

The Destroy function removes the specified OneCatalog instance from the system.

The GetCollection function gets a group of data items from the catalog database in accordance with predefined selection criteria provided to the function.

The GetItem function gets the contents of a OneCatalog item specified by the catalog ID.

The ModifyCatalog function modifies the properties of the specified OneCatalog section.

The RemoveItemFromCatalog function removes the specified item from the specified catalog section.

The RetrieveCatalog function retrieves the specified OneCatalog and places it in the RESPONSE section of the session. The request may specify how many levels of the catalog to retrieve.

The OneChat application component allows site users to discuss topics synchronously in multiple channels through a Java client. In the illustrative embodiment, the OneChat service can include a OneChat ACT, a Windows NT server process, and a Java applet.

OneChat differs from most components in that it implements a downloaded client application providing the following functionality, through an application component and not through a service component. Thus, active content instances are allowed for ACTs as well as passive content instances.

Channels - the ability for a user to participate in multiple ongoing chat channels.

Private chat - the ability for two users to create and use a private chat channel.

Invite - the ability to invite other users to join in the chat channel.

Join and Leave - the ability to enter into and exit from a chat channel.

Kick - the ability to disconnect a user from a chat channel; reserved for administrators.

Buddy Support - the ability to see the status of selected "buddy" users.

The OneEvent application component allows the display and management of any community event, e.g., a bake sale or car wash, within the system.

The OneInbox application component provides a means for storing work items and assigning them to workstation users. A work item is a reference to another ACT

item such as an Order or Inquiry, which is in need of intervention by the workstation user. The user checks his or her inbox for work items upon logging into work station.

It is important to understand that the "Inbox" is not an inbox in the sense of a place to store message or deliver email. It is a tool to assign application component instances that require external processing to a user who can perform that processing.

The Inquiry Application Component provides customer inquiry processing functionality for customer service and support. The Inquiry component application enables efficient management and tracking of the inquiries generated by the customers. The following provides examples of the methods services and functions, both standard interface and component-specific, that can apply to OneInquiry component.

AddInquiry creates a new inquiry and adds it to the inquiry queue for later processing.

Create is used to create an inquiry instance such as a question about a hard good product. The OneInquiry application component is responsible for generating a unique key (probably through a function in the utility module) and returning it to the caller in the idItemID parameter. The Create function sets up the mandatory fields of an item. Other elements have to be set individually via the Set method.

Destroy deletes the product item from storage.

GetInquiryID returns the unique identifier of the Inquiry.

GetInquiryStatus returns the status of the inquiry specified.

GetInquirySubmissionMethod returns the inquiry SubmissionMethodID.

GetInquiryID returns the unique identifier of the Inquiry.

RemoveInquiry removes an inquiry from the queue.

GetInquiryID returns the unique identifier of the Inquiry.

The OneLink application component allows a site to host a list of internet links which can be categorized into one or more hierarchical structures, similar to the directory structure provided by the Windows Explorer interface.

The OneNotes application component provides the ability to allow users to attach internal notes to application component instances (ACIs). The notes are intended for site-internal use only, allowing a business to maintain ad hoc information about customers, orders, or inquiries.

The OneOrder application component manages stored data relating to customer orders. The OneOrder service component manages Internet order creation, processing, and reference functionally. The following provides examples of the methods services and functions, both standard interface and component-specific, that can apply to this ACT.

AddToOrder adds an order item to an order in the specified session. It checks for an OrderID session variable and creates a new order item that references the OrderID session variable in the OrderItems table. If there is no OrderId in the specified session, the OneOrder SCT creates a new order in the Orders table, and creates a new order item that references the OrderId in the OrderItems table.

CalculateSubtotal calculates the subtotal of all order items from an order in the specified session. OneOrder checks the session for an OrderID and sums prices of all items in the order, multiplied by quantity, to return a total order price.If there is no order specified in the session, a subtotal of zero is returned. A new order will NOT be created.

The Create method creates a new OneOrder instance and initializes it with the values specified in an xmlOneOrder parameter. Create returns a OneOrder IDTYPE structure.

DeleteOrder deletes the order from the specified session. OneOrderSvc checks the session for an OrderID, deletes that order and order items from the Order and OrderItems database and removes it from the session.

The Destroy method removes the specified OneOrder instance.

The GetCollection method gets a collection of all orders.

The GetItem method retrieves all or specified elements of the OneOrder instance specified in the XML structure xmlOneOrderId.

GetOrder lists the order items from an order in the specified session. OneOrderSvc checks the session for an OrderID and creates and returns an XML order string that represents the order. If there is no order specified in the session, an XML order string with no items is returned.

ListOrders lists previous orders placed by the user.

OneOrder checks the session for a non-guest user id, searches the OneOrder ACT for old orders owned by this user, and returns an XML order list string that represents all known orders. If no orders are found, OneOrder returns an XML order list string with no orders. This method can require authentication for use.

OrderStatus returns the status of an order. OneOrder checks the session for an OrderID and determines the status of the order specified by OrderID.

Purchase processes the purchase of the order in the specified session. OneOrder checks the session for an OrderID and executes Tax and Shipping computations,

processes the purchase with the Payment connector, and changes the orders' status to purchase.

This method requires authentication for use.

RemoveFromOrder removes the specified quantity of an order item from an order in the specified session. The OneOrder service component checks the session for an OrderID and removes the specified order item amount from that order. If Quantity is zero or greater than the total quantity, the whole order item and quantity are removed from the Order and OrderItems tables.

The SetItem method sets values in the specified OneOrder instance.

Split splits a single order into two, separately managed orders, each with unique order Ids. This method can require authentication for use.

The OneProduct application component manages creation, storage, and retrieval of hard good product offerings.

OneSession - the OneSession application component manages storage and retrieval of customer session information. It associates information about a user's actions during a session with the session. This data may be used by other service components or other services to make decisions based on the user's behavior, or to update customer profile information. The business model controls when session information is captured and destroyed. The following provides examples of the methods services and functions, both standard interface and component-specific that can apply to this component.

BeginRequest retrieves previously saved information from the OneSession ACT and restores the user's session. It does not require any data as input, but any data present will be copied to the output with the saved data.

Create creates a new OneSession instance and initializes it with the values specified in the OneSessionIn parameter. It returns the OneSession instance identifier. A OneSession instance may also be referred to as a "session profile."

Destroy removes the specified OneSession Profile instance.

EndRequest is called at the end of a request to save the current state of the session to the database.

EndSession changes the status of the specified session profile to TERMINATED and fills in all associated properties. This method can only be called on session profiles that are not already terminated; if the session has already been terminated, an error is generated.

GetCollection is not implemented for this release of the OneSession component.

GetItem gets the contents of the session profile specified by the section ID.

SetItem updates the contents of the specified session profile with new values.

Presentation Components

Presentation components provide view generation services that are a special type of data transformation service. In accordance with the present invention, the presentation components take in XML data and produces data in a client viewable format such as, for example, an HTML formatted web page viewable through a web browser. A view generation service is typically one of the last services in a Business Model since it changes the data structure from one which is useful for understanding the data to one that is useful for formatting or presenting the data to the customer in a client application such as a web browser.

All of the requests that a customer makes from his initial request until the customer explicitly logs out or times out due to a period of inactivity can be tied together into a single session. The system 100 can start the recording of a session at the customer's initial request even before the customer has been authenticated. The system can use the information obtained prior to authentication to impact how the controller implements a business model. Preferably, every action that a customer can take within a session is controlled by a business model that is valid for the customer. It is therefore important to keep track of every action that the customer takes. For example, if any of the services rely on information which was generated by previously executed services, then this information must be stored and restored when the customer makes subsequent requests. This function can be provided the OneSession ACT. The OneSession ACT restores the customer session information to the XML data structure for the session every time it receives a request for a web page which invokes a business action. The system 100 stores the session data for each customer in the OneSession ACT.

In processing a request, the controller may call on a number of services to complete its task and each service function may use any number of data items (ACT data) in performing its service. Therefore, a single request may have many data items associated with processing the request and the system can group all or some of the data items into one larger data envelope that can be passed to each service used in processing the request. The services can use whatever data is needed and optionally add new data to the envelope, remove data from it, or modify existing data in it.

Preferably, each request performs only a single business action in the business model that defines the operation of the site and may rely on data that was generated on

previous requests. The state of the business model for each customer can be preserved across each of the requests. For example, an order is created the first time that a customer adds an item to his shopping cart. The same order is then updated each time the customer adds or removes an item and is processed when the customer checks out. The identifier of the customer and the order must be maintained between each request to add an item, remove an item, checkout, or initiate any other action.

In one embodiment, a session ACT, such as OneSession, is provided to save and restore the session data in the session data envelope. The session data envelope can hold any of the data items that a given service may need to use or add to the session data item or alternatively, it can provide a pointer to a database or data item that a given service can use. This data can be used by other services later in the business action or model or by any service called in a subsequent request within the same session. The session ACT saves the current state of the session data envelope at the end of one request and restores it at the beginning of the next request.

Preferably, at the start of each request, the session ACT BeginRequest function is called. This function has no input data; it restores the session data based on the session identifier received from the customer's web browser's (such as from an HTML form or a "Cookie"). The session data is restored from the session ACT which holds whatever data was left in the session item when the previous request of the current session ended. At the end of the request, the session ACT EndRequest service can be called to save the data in persistent storage via the Session ACT. Services that need to maintain state within a session can use this ACT to hold their state information by inserting or modifying the fields in the XML of a particular session data item of the session data envelope. The

identifiers of any ACT items that are inserted into the session can be stored so that later functions can restore this information as needed.

In one embodiment, one particular ACT item identifier that is always present is the Customer item identifier. The Customer item identifies the customer who is using the session and becomes useful for profiling the customer after the customer has logged in. A unique but 'anonymous' Customer item can be used in the first request of a session to make sure that each session has a customer associated with it.

Figure 5 shows an example of a typical Session ACT data item. The data items that it contains, such as the Customer and Order data items, for example, can be saved and restored during the course of the session. Other services can obtain the full or partial contents of the session data item from the appropriate session ACT and can insert additional data into the session data to be held for the duration of the request or session. The data that is added or modified during the session can be provided in any valid XML data format and the system 100 will save and restore the data over the course of one or more sessions.

Request Data

The Session Service Component BeginRequest service is also responsible for retrieving information from the request (such as information provided by a CGI script, ASP script or Visual Basic script), converting it to XML and inserting it into the Session data item so it is available for all of the services which are called during this request. This information can include: any data that was entered by the customer in a form, any variables on the URL, the customer's browser and local host information, information about the server and requested URL, and cookie and certificate information. Figure 6 shows request data which can be obtained from the web server variables, converted to

XML and placed in the Session data item. This information can be available to all services invoked during a specific request.

Some services utilize information concerning functions that have been performed by services that were called previously in the session. This is particularly true in the case of the services offered by the Audit Service Component which are responsible for saving this information and converting it into easily understandable summary data for performance monitoring and reporting services.

In one embodiment of the invention, every service component can add a description of each service (function) that it performed as well as any additional information, which the Audit Service Component can log, in the session data stored by the OneAudit ACT. For example, this information can include the following:

TYPE - The name of the service component

ACTION - The name of the service function which was performed

INFO - Optional information that further describes the service

END_TIME - Day and time that the service completed

STATUS - 'Success' or an error code indicating the type of failure that occurred

ACT - Similar information for each ACT function that was called by the service function

A utility function can be provided to help service component developers to develop SCTs that add audit information to the session data. A function can also be provided to automatically add the ACT data to the audit information. Figure 7 shows an example of an Audit data item which was added to the session data by the Order Service Component while performing an AddToOrder.

The Controller and Business Models

The controller 110 controls the customer flow through the system 100 which can present to the customer an e-commerce or online store. The nature of the world wide web allows a customer to have considerable freedom in determining where to go and what to do next by the links they select. This is accomplished by providing the customer with many links to choose from in response to each requested action. However, the customer can go outside the links provided by making requests by typing in a URL directly, or by selecting a book marked URL. In one embodiment, the Controller sits between the web server, which dispatches the request to be processed, and the services that are used to perform that action and generate the responsive web page. This way, the controller can make sure that no actions are performed outside of what is allowed by the customer or is appropriate for the current state of what he has done.

The Controller does this by applying a particular Business Model to every request that the customer makes. The Business Model governs what actions a customer may perform and what he must do prior to performing that action.

The Controller

The controller operates to ensure that all of the customers actions fit within a predefined business model. The controller processes an action requested by a customer according to a business model defined for that customer or situation. As one having ordinary skill will appreciate, the relationship between the control component and the business model can vary such that a tightly integrated control component - business model based system can be used in an optimized and highly efficient system, requiring only limited functionality, whereas a more structured and clearly separated control

component - business model based system can be used when flexibility is valued more than efficiency. Both the control component and business model programming make up the code that can be used to execute each business action. However, as shown in FIGURE 3, whether the business model information is hard coded or created dynamically, preferably, every business action includes the following information:

customer access privileges 330a that specify which customers may execute the business action,

prerequisite business actions 330b that must have been successfully completed by the customer before this business action can be executed,

required input data 330c that must be provided by the customer when requesting to execute the business action,

required content 330c which the business action uses during execution,

processing logic 330d which controls what the business action does,

output display format 330e for the data which is generated by the business action and will be displayed to the customer, and possible next actions that a customer may choose from.

Business Action Level Access Control

The controller will utilize an SCT, such as an access service component, to determine if the customer has access to the requested action. If a customer attempts to perform an action that he is not permitted to perform, then the controller can be provided with a number of alternative actions or functions. The controller may give the customer an opportunity to upgrade his identity (e.g. guest customers can login as a registered customer, already registered customers may re-login as an administrative customer)

Alternatively, the controller may process a similar action that the customer does have access to or let the customer choose from a list of alternative actions. If nothing else, the controller can report to the customer that the customer does not have the proper privileges to perform the requested action.

Required Inputs

If the customer has not supplied all necessary inputs to perform the action, then the controller can redirect the customer to a form, such as an HTML form, that can be used to supply the missing information. Preferably, this form should be pre-filled with all of the information that the controller already knows and indicate which of the missing information is required. The customer can then modify the information, add to it any missing information, and resubmit the request. The customer can also choose to cancel the request either explicitly by clicking on a cancel button on the input form or implicitly by not supplying the additional information within a specific, allowed period of time.

Data Access Control

Many business actions require access to information in addition to that input by the customer in order to proceed. For example, searching a product set, viewing product information, viewing customer information, viewing the contents of a catalog, require access to product, customer or catalog ACTs. This data may or may not be access controlled. The controller can utilize an SCT, such as an access service component, to determine if the requesting customer has been granted access to all data that is needed to perform the requested action. If the customer has not been granted sufficient access, then

the controller may direct the customer to a different action or return an error message.

Prerequisite Actions

Prior to initiating any business action, the controller can compare the prerequisite actions associated with a given business action with a log of the prior actions completed by the customer that is stored in an audit log or session log. The logging of prior actions completed can be performed by an SCT, such as a session service component, that tracks action completion and uses an ACT, such as a session log application component, to store a history of completed actions. If the customer has not performed a prerequisite action, then the controller can attempt to execute the prerequisite action instead. After the prerequisite action or actions are executed, the controller can reinitiate the prior action.

Business Model

The controller utilizes a business model in making all decisions. Specialized business models can be provided to allow turnkey installation for some common e-commerce applications. One example is a shopping controller which can include a business model that allows a customer to interact with a site directly through a catalog. The shopping controller can provide personalized presentation of dynamic catalog content to a customer. The customer can move through the sections of a virtual store, view detailed representations of products and

services (hard-goods, digital content, chat rooms etc.) in an intuitive way. The client is then charged for all selected items purchased or services used. Some items or services may be free to customers.

In one embodiment, the system can include a predefined Business Model that incorporates a Targeting Controller. The Targeting Controller can include a business model that allows a customer to receive a completely personalized, targeted product and service offering from the virtual store. The Targeting Controller can utilize information provided by the customer as well as historical information about the customer's past buying habits and web pages viewed to create a customer profile, identifying the customer's interests. The Targeting Controller can use the customer profile to dynamically generate targeted product and service offerings during the customer session. This business model can reduce the amount of information that a customer needs to grapple with and increase the likelihood of making a sale before the customer goes to another website. The customer can also see detailed representations of any of the targeted hard-goods, digital content, chat rooms, or other items that might be of interest to the customer in an intuitive way and provide compensation (such as discounts) in appropriate ways for all items being purchased. This enables marketing information to be used effectively.

According to a preferred embodiment, a system according to the invention provides tracking and profiling as described below. The OneMeta ACT manages the meta-data categories and values which the system uses for targeting and profiling. It maintains a simple data structure such as that presented in the following table.

Meta-Data Category	Meta-Data Category Element
Age-Group	Senior
Age-Group	Teenager
Clothing-Type	Shoe

Material	Glass
Material	Leather
Interest	Sports

The OneUpSell ACT manages recommended upsell item(s) for any ACT instance. The OneUpSell ACT includes information such as that presented in the following table.

ACT	ID	Recommend ACT	Recommend ID
Product	11	Product	300
Product	11	Chat	82
Chat	4	Product	5
Product	7	Chat	12

The OneCrossSell ACT manages recommended upsell item(s) for any ACT instance. The OneCrossSell ACT includes information such as that presented in the following table.

ACT	ID	Meta-Data	Recommend ACT	Recommend ID	Recommend Meta-Data
Product	11		Product	300	
		Clothing-Type/Shoes			Clothing-Type/Socks
Chat	4		Chat	5	
Chat	4		Product	84	
Product	7				Interest/sports

The OneTarget ACT manages the meta-data tags associated with each ACT instance in the system (that has been tagged by our customer). The OneTarget ACT includes information such as that presented in the following table.

Application Component Type (ACT)	Instance ID	Meta-Data Tags
One Product	11	Age-Group/teenager, Clothing - Type/Shoe, Material/leather, Interest/sports
OneChat	6	Clothing-type/shoe
OneConference	12	Age-Group/teenager, Interest/sports
OneProduct	8	Age-Group/teenage, Material/leather

The OneProfile ACT tracks customer affinities for meta-data categories and meta-data tags. The OneProfile ACT includes information such as that presented in the following table.

User ID	Meta-Data Tag	Cumulative Affinity Points	Relative Customer Affinity
11	Material/Glass	10	10/267 = 03.745 %
11	Age-Group/Teenager	58	58/267 = 21.723 %
11	Clothing - - Type/Shoe	115	115/267 = 43.071 %
11	Age Group/Senior	1	1/267 = 00.375 %
11	Material Leather	11	11/267 = 04.120%
11	Interest/Sports	72	72/267 = 26.966%
11	TOTAL:	267	
11	Material/*	22	22/267 = 8.240%
11	Age-Group/*	59	59/267 = 22.097%
11	Clothing-Type/*	115	115/267 = 43.071%
11	Interest/*	72	72/267 = 26.966%

The OneProfile SCT analyses the click-trail executed during a customer session (which is held in the OneSession ACT) to score the affinity of that customer session with meta-data tags placed on Application Component Type Instances touched during the click-through. This process works as follows. Assume the OneSession ACT contains the click trail information listed below for a customer session.

Application Component Type (ACT)	Instance ID within the ACT	Number of Clicks
OneProduct	11	5
OneChat	6	10
OneConference	12	2
OneProduct	8	1

Assume the OneTarget ACT contains the meta-data tags listed below for the referenced ACT instances drawing on the OneMeta ACT to select meta-data categories and meta-data tags within categories:

Application Component Type (ACT)	Instance ID	Meta-Data Tags
One Product	11	Age-Group/teenager, Clothing-Type/shoe, Material/leather, Interest/sports
OneChat	6	Clothing-type/shoe
OneConference	12	Age-Group/teenager, Interest/sports
OneProduct	8	Age-Group/teenager, Material/leather

Thus, with the given click-trail and targeting information OneProfile can extract the following affinity vector for this session:

Meta-Data Tag	Meta-Data Category Element
Age-Group/Teenager	5 + 0 + 2 + 1 = 8
Clothing Type/Shoe	5 +10 + 0 + 0 = 15
Material/Leather	5 + 0 + 0 + 1 = 6
Interest/Sports	5 + 0 + 2 + 0 = 7
TOTAL:	This is 36 total affinity points

OneProfile can now draw the following conclusions about relative affinity this Session:

Meta-Data Tag	Relative Affinity This Session
Age-Group/Teenager	8/36 = 22.222%
Clothing Type/Shoe	15/36 = 41.667%
Material/Leather	6/36 = 16.6667%
Interest/Sports	7/36 = 19.445%
TOTAL:	This is still 36 total affinity points

The system can assume that this customer's previous affinity profile, stored in the OneProfile ACT is as follows:

Meta-Data Tag	Cumulative Points Affinity	Relative Customer Affinity
Material/Glass	10	10/231 = 04.329%
Age-Group/Teenager	50	50/231 = 21.645%
Clothing-Type/Shoe	100	100/231 = 43.290%
Age-Group/Senior	1	1/231 = 00.433%

Material/Leather	5	5/231 = 02.165%
Interest/Sports	65	65/231 = 28.139%
TOTAL:	231	

UPDATE

The update method in the OneProfile SCT takes a customer session as input and updates this customer's affinity matrix in the OneProfile ACT. In this example, updating this customer's affinity matrix with this session results in the following new affinity matrix:

Meta-Data Tag	Cumulative Points Affinity	Relative Customer Affinity
Material/Glass	10	10/267 = 03.745%
Age-Group/Teenager	58	58/267 = 21.723%
Clothing-Type/Shoe	115	115/267 = 43.071%
Age-Group/Senior	1	1/671 = 00.375%
Material/Leather	11	11/267 = 04.120%
Interest/Sports	72	72/267 = 29.966%
TOTAL:	267	Approximately 100%
Material/*	22	22/267 = 8.240%
Age-Group/*	59	59/267 = 22.097%
Clothing-Type/*	115	115/267 = 43.071%
Interest/*	72	72/267 = 29.966%

Observe that this affinity matrix is stored not only by meta-data tag but also by meta-data category to support the targeting SCT's operations. (See the last four rows).

In one embodiment, the system renews customer profiles over time. Otherwise, the law of big numbers will introduces an averaging of affinity across all meta-data tags.

The OneProfile SCT, therefore, offers a renew method that takes two parameters indicating how much the profiles should be renewed and which profiles are old enough to merit renewing. A call to renew *(x,y)* with a large *x* wipes away more of a historical pattern and makes the profile more sensitive to current activities. A call to renew *(x,y)* with a smaller value for *x* allows current clicks to have a significant impact while including historical clicks in the calculation of affinity. The *y* parameter indicates which profiles are old enough to renew. Only customers whose total affinity point value is great than *y* will be affected by the renew operation.

If total affinity points < = y

then skip this customer's profile in the renewal process

Else if (Cumulative Affinity Points) x) > = 1

then Cumulative Affinity Points = (Cumulative Affinity Points) x)

else remove all trace of any affinity for this meta-data tag from the customer profile.

The OneTarget service component examines the relative customer affinities for meta-data tags and selects those ACT instances, which statistically are appealing. This can be done in a variety of ways.

To compute the absolute preference that a customer has for an ACT instance, the system sums his/her affinities for each meta-data tag on that ACT instance. Consider the above customer profile and the following application component type instance.

Application Component Type (ACT)	Instance	Meta-Data Tags
OneProduct	84	Age-Group/teenager, Interest/sports, Material/glass, Material/leather

The relative preference for this content instance is computed by summing the relative preferences for each tag. In this case the result is:

Age-Group/teenager 21.723% + Interest/sports 26.966% + Material/glass 3.745% + Material/leather 4.120% = 56.554%.

The absolute preference method in the OneTarget SCT should return a set of ACT instances in decreasing order of absolute preference. The system attaches a pair of parameters here where the first indicates a threshold of absolute preference and the second indicates a maximum number of instances to return. Passing in a 0 for either or both parameters indicates no bound.

Absolute Preference (percent threshold, instance count threshold)

In the running example here, this customer seems to care most about clothing-type, then interest, then age-group, and finally least about material. These affinities demonstrate the customer's preference by meta-data category. The CatagoryPreference method in the OneTarget SCT performs a weighted average of affinity based on the meta-data tag's category. In this example the weights for this customer are: clothing-type (4), interest (3), age-group *(2),* material (1).

*Age-Group/teenager 27.723 * 2 + Interest/sports 26.966 * 3 + Material/glass 3.745 * 1 + Material/leather 4/120 * 1 = 55.466 + 80.898 + 3.745 + 4.120 = 144.229/7 = 20.604*

A high affinity for a particular value (say Interest/sports) has more impact if the Interest category frequently draws this customer's click and less impact if the Interest category is

not so compelling. Once again, the system uses the two parameters for percent threshold and absolute number of instances threshold with 0 in either/both places removing the restriction.

CategoryPreference percent threshold, instance count threshold)

OneTarget has 2 more ACTs beneath it (beyond the meta-data tags on ACIs held in the OneTarget ACT). One ACT is called OneUpsell ACT and stores which particular ACT instance should be pushed either as an up-sell to some other ACT instance. The OneTarget SCT has a method called Upsell that takes an ACT/ID pair and responds with a set of ACT instances to up-sell. It cuts off this list at threshold elements in the response. If 0 is passed in as the threshold then the number of instances in the response is not limited.

Upsell (ACT, ID, threshold) returns the ACT instance(s) that we recommend upselling

The other ACT OneTarget has beneath it is called OneCrossSell ACT and stores which ACT instances should be pushed as a cross-sell to some other ACT instance or meta-data tag. The OneTarget SCT has a method called CrossSell that works like this:

CrossSell (ACT, ID, Meta-Data, threshold) returns the ACT instance that we recommend cross-selling

The method can be invoked with either an ACT/ID pair or a Meta-data tag along with a threshold for the maximum number of instances to return and the method responds

with a set of recommended ACT instances to cross-sell. If 0 is passed in as the threshold then the number of instances in the response is not limited.

Another business model can be a Searching Controller. The Searching Controller can include a business model that allows a customer to receive an offering of items from a virtual store that match the customer's search criteria. This business model reduces the amount of information that a customer needs to grapple with and increases the likelihood of making a sale. It also gives the customer a certain sense of control. Once the products sought are found, the customer can see detailed representations of these hard-goods, digital content, chat rooms, and other items in an intuitive way and provide compensation in appropriate ways for all items being purchased. This business model can be used to appeal to and retain those customers that want to control their own destiny in a virtual store without having to walk through it section by section.

Workstations

Workstations can provide an administrator's view of data contained in ACTs or generated by SCTs. Workstations can also provide a customer's view of the same data. A workstation can contain business logic and control of its own or it can rely on the services and a business model. Any logic that is used in a service could be useable by other workstations and components.

A Workstation installation package includes the following: Documentation; List of required ACT and SCTs; ASP files to handle views; and XSL views for all data.

A Workstation can provide a view on top of ACTs and SCTs and can include external ASP, HTML, etc. A Workstation can also provide pure views with no business logic.

The e-commerce system can include a higher level monitoring and reporting system for collecting data about the operation of the transaction processing system over a period of time and reporting information relating to the performance of the system. The system can include sales channel modeling and reporting which is described in further detail in Appendix C. In addition, the system can include technical modeling and reporting as well.

Administrative functions are handled by role-based workstations. Role-based administrative workstations allow a company to distribute the responsibility for various administrative functions to any of the available administrative roles (such as content manager, marketing manager and channel manager). This provides the system with complete flexibility to distribute and redefine responsibilities as the developer requires.

In one embodiment, the e-commerce system can be adapted to run on a distributed processing system or server cluster, such as a Distributed Internet Server Array (DISA) available from Compaq Computer Corporation, Houston, Texas. This hardware platform provides scaleability by allowing server resources to added as demand increases. As described in Appendix E, the system can provide both front-end and back-side load balancing to distribute the front end and back-side server loads to optimize server resources.

Financial Modeling

A preferred embodiment of a system according to the invention provides financial modeling for a developed e-commerce solution. The following is a simple, mathematical framework that quantifies performance in a sales channel. The model is presented in three tables. The first defines the variables involved. The second presents

business values that can be computed over the variables. The third table presents retained net profit. This model demonstrates the system's ability to improve the customers' retained net profit.

Variables

Variable	Definition
C	Average number of Customers
V	Average number of Visits per customer
P	Average number of customer contacts Per visit
O	Average number of Offers per contact
TO	Average number of Targeted Offers; offers of interest to the end customer
EO	Average number of Executed Offers; offers with agreed, executed terms of purchase
RO	Average number of Retained Offers; offers not returned or rejected after execution
OC	Average Offer Cost; cost of extending an offer to a customer
EC	Average Execution Cost; cost of executing an offer on behalf of a customer
RC	Average Return Cost; cost of accepting a return on behalf of a customer
CC	Average Channel Cost; amortized cost of building and maintaining the channel
UP	Average Unit Price
MP	Average Margin as a Percent of unit price
SP	Average number of Satisfactory customer contacts Per visit

Business Functions

Function		Equation
TotalOffers	=	C*V*P*O
Targeted Offer Percentage	=	TO ÷ Total Offers
Executed Offer Percentage	=	BO ÷ Total Offers
Retained Offer Percentage	=	RO ÷ Total Offers
Number of Returns	=	EO-RO
Executed Sales	=	EO * UP
Retained Sales	=	RO * UP
Executed Gross Profit	=	Executed Sales * MP
Retained Gross Profit	=	Retained Sales * MP
Retained Net Profit	=	Retained Gross Profit - (Total Offers * OC) - (EO * BC) -((EO - RO) * RC) - CC
Return on Investment	=	Retained Net Profit + ((Total Offers * OC) + (BO * BC) + ((BO-RO) * RC)- CC)
OneSoft Profit Growth Factor	=	Retained Net Profit (Current System) ÷ Retained Net Profit (Reference System)
Return Comparison between channels	=	Retained Net Profit (Current System) / Retained Net Profit (Reference System)
Lifetime Value of the Customer Relationship in the Channel	=	SP÷P

Profit/Return Equation

The system can formulate a profit equation that ties all of the key variables together into a final number representing cash flow. The function is defined using the business functions above. In the table below, the system substitutes all the variables and presents the final equation.

Complete Retained Net Profit is:

(Retained Offers * Average Unit Price * Average Margin as a Percent of Unit Price) - (Total Offers * Cost of Extending an Offer) - (Number of Executed Offers * Cost of Executing an Offer) - (Number of Returns * Cost of Processing a Return) - Amortized Channel Cost

Complete Retained Net Profit Equation
(RO*UP*MP)-(C*V*P*O*OC)-(EO*EC)-((EO-RO)*RC)-CC

The virtual channel, selling on the Internet, is attractive because it minimizes many of the costs incurred by doing business in other ways. Below the channel comparison function is applied to analyze the difference between Internet business and typical brick-and-mortar retail selling.

Compute Retained Net Profit (Current System) + Retained Net Profit (Reference System) over the first year with the following values for each of the necessary variables. The illustration below makes assumptions including what percentage of total offers are targeted, which percentage of those are executed, and which percentage of those are retained. The illustration also makes an assumption about what percentage of visits are customer satisfying in the online and in-store cases. This illustration assumes values in the context of a very successful in-store retailer taking full advantage of personalization mechanisms to be customer-centric on the Internet. The assumptions are indicated in the first few rows of the table.

Retained Net Profit Equation:	**Internet with OneSoft**	**Brick and Mortar Retail**
(RO*UP*MP)-(C*V*P*O*OC)-(EO*EC)-((EO-RO)*RC)-CC		
Targeted Offer Percentage	70%	60%
Executed Offer Percentage	15%	5%
Retained Offer Percentage	90%	80%
Satisfying Customer Contact Percentage	98%	80%
Retained Sales Revenue (RO*UP*MP)	$141,750,000	$720,000
Cost of Making Offers (C*V*P*O*OC)	$60,000	$80,000
Cost of Executing Offers (EO*EC)	$90,000	$8,000
Cost of Handling Returns ((EO~RO)*RC)	$90,000	$2,400

Cost of Selling up the Channel (CC)	$200,000	$500,000
C	50,000	10,000
V	5	2
P	4	2
O	6	2
TO	4,200,000	48,000
EO	900,000	4,000
RO	810,000	3,200
OC	$0.01	$1.00
EC	$0.10	$2.00
RC	$1.00	$3.00
CC	$200,000.00	$500,000.00
UP	$500	$500
MP	35%	45%
SP = Long Term Customer Value	980,000	32,000
Net Profit	$141,310,000	$129,600

In tabular form, this illustration indicates which elements most directly relate to the customer's profits within the virtual channel.

The table below indicates which variables system components address favorably.

Product and Service Lines	**C**	**V**	**P**	**O**	**TO**	**EO**	**RO**	**OC**	**EC**	**RC**	**CC**	**UP**	**MP**	**SP**
Internet Operations Center								*	*	*	*			*
Implementation Services								*	*	*	*			*
Virtual Channel Strategic Services	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Graphic/Multimedia Design Services	*	*	*	*		*	*			*	*	*	*	*
Community Application Components	*	*	*	*	*	*	*			*		*	*	*
Catalog Application Components					*	*	*	*	*	*	*	*	*	*
Customer Service Application Components	*	*	*	*		*	*		*	*		*	*	*
Offer-Targeting Service Components	*	*	*	*	*	*	*		*			*	*	*
Search Service Components	*	*	*	*	*	*	*			*		*	*	*
Customer Profiling Service Components	*	*	*	*	*	*	*			*		*	*	*
Virtual Store Design Service Components	*	*	*	*	*	*	*				*	*	*	*
Legacy System Connector Components	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Payment System	*	*	*	*		*	*		*	*	*	*	*	*

Connector Components														
Fulfillment System Connector Components	*	*	*	*		*	*		*	*	*	*	*	*
Business Model Components	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Marketing Components	*	*	*	*	*	*	*			*	*	*	*	*
System Monitoring components	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Sales Management Work Station Components	*	*	*	*	*	*	*		*	*	*	*	*	*
Application Management Work Station Components	*	*	*	*	*	*	*	*	*			*	*	*
Marketing Strategy Work Station Components	*	*	*	*	*	*	*	*	*	*		*	*	*
Business Modeling Work Station Components	*	*	*	*	*	*	*	*	*	*	*	*	*	*
System Monitor Work Station Components	*	*	*	*	*	*	*	*	*	*		*	*	*

One of the advantages of a system in accordance with the present invention is that each of the components are functionally independent, and thus do not require knowledge about other components to perform their function. This allows complex systems to constructed the same way simple systems are constructed, by adding components in a "plug and play" fashion. Thus, standard e-commerce system types can be built as objects that can form the components of larger systems.

The invention may be embodied in other specific forms without departing from the spirit or essential characteristics thereof. The present embodiments are therefore to be considered in respects as illustrative and not restrictive, the scope of the invention being indicated by the appended claims rather than by the foregoing description, and all changes which come within the meaning and range of the equivalency of the claims are therefore intended to be embraced therein.

A Modular System For Processing Network Based Transactions

What is claimed is

1. A system for processing a transaction comprising:

a computer processing system and associated memory;

a plurality of independent component modules operatively coupled to the computer processing system, wherein each module is selected from a group of modules consisting of

application component modules adapted for providing data management functions;

service component modules adapted for providing data transformation, monitoring, and commerce, functions;

presentation component modules adapted for providing data presentation functions; and

control component modules adapted for communicating with each of said independent modules for processing said transaction;

wherein each of the independent modules is adapted for providing at least one predefined function without the assistance of any other of said independent modules.

2. The system of claim 1 wherein the system includes a control component module adapted for communicating with at least one of the plurality of independent modules to process the transaction.

3. The system of claim 1 wherein each of the plurality of independent modules is adapted for communicating with any other independent component module according to a common extensible interface.

4. The system of claim 3 wherein the extensible interface comprises means for transmitting and receiving a data record according to an extensible protocol.

5. The system of claim 3 wherein each of the plurality of independent component modules is adapted for communicating with any other independent component module according to extensible data format.

6. The system of claim 5 wherein the extensible data format includes an extensible markup language.

7. The system of claim 1 wherein the system comprises an application component module adapted for providing data management functions.

8. The system of claim 7 wherein the data management functions include creating, deleting, retrieving, and modifying data.

9. The system of claim 7 wherein said application component module is an audit component module adapted for managing data generated as a function of monitoring and logging activities associated with said transaction.

10. The system of claim 7 wherein said application component module is a catalog component module adapted for managing data representative of a collection of products.

11. The system of claim 7 wherein said application component module is a communication component module adapted for managing data representative of communication information.

12. The system of claim 7 wherein said application component module is an event component module adapted for managing data representative of an event.

13. The system of claim 7 wherein said application component module is an orders component module adapted for managing data representative of customer orders.

14. The system of claim 7 wherein said application component module is adapted for managing data representative of customer session information.

15. The system of claim 7 wherein each of said plurality of application component modules provides functions which render an instance of an application component representative of data managed by said application component.

16. The system of claim 7 wherein said application component module provides data management functions in response to a request from a service component module.

17. The system of claim 1 wherein the system comprises a service component module adapted for providing a data transformation function.

18. The system of claim 17 wherein the data transformation function is a data filtering function.

19. The system of claim 17 wherein the service component module is adapted for providing a data monitoring function.

20. The system of claim 17 wherein the service component module is adapted for a data logging function.

21. The system of claim 17 wherein the service component module is adapted for providing a customer shopping basket function.

22. The system of claim 17 wherein the service component module is adapted to provide customer order services.

23. The system of claim 17 wherein the service component module is adapted to provide a customer access function.

24. The system of claim 17 wherein the service component module is adapted for communicating with at least one application component module.

25. The system of claim 1 wherein the system includes a presentation component module adapted for converting data to a data presentation format.

26. The system of claim 25 wherein the data presentation format is a hypertext markup language.

27. The system of claim 25 wherein said presentation component module is adapted for receiving data in an extensible data format and merging said received data with at least one presentation template in order to produce data in said data presentation format.

28. The system of claim 27 wherein said extensible data format is an extensible markup language (XML); said presentation template is an extensible stylesheet language and said data presentation format is HTML.

29. The system of claim 1 wherein the system includes a control component module adapted for controlling at least one of said independent component modules for processing said transaction.

30. The system of claim 29 wherein said system includes a business action and business action defines which of said plurality of independent component modules are to be used by said control component module to process said transaction.

31. The system of claim 30 wherein said business action is defined in a dynamic web page format and said plurality of independent component modules are objects invoked by

said control component module as a function of information provided by a participant of said transaction.

32. The system of claim 31 wherein said dynamic web page is an Active Server Page (ASP) format and independent component modules are Component Object Model (COM) component modules.

33. The system of claim 29 wherein the business action is integrated with the control component module to provide one or more predefined business actions.

34. The system of claim 29 wherein the control component module is adapted for determining whether prerequisite actions are required before beginning a business action.

35. The system of claim 1 comprising a shopping controller including the control component module and a shopping business model adapted for allowing a user to interact with a catalog through a web site, and paying for items purchased.

36. The system of claim 1 comprising a shopping controller consisting of the control component module and a shopping business model, said shopping controller being adapted to allow a user to interact with a catalog through a web site and pay for items purchased.

37. The system of claim 1 comprising a searching controller, including the control component module and a searching business model, said searching controller being

adapted to allow a user to submit a search request of a list of items and to receive information that matches the user's search request.

38. The system of claim 1 further comprising connector component modules adapted for communicating with an external computer processing system to transmit data to or receive data from said external computer processing system.

39. The system of claim 38 wherein said connector component module is adapted for converting date received from said external computer processing system to a standardized data format.

40. The system of claim 38 wherein said connector component module is adapted for converting date sent to said external computer processing system to a non-standardized data format used by said external computer processing system.

41. A system for processing a transaction comprising:

a computer processing system and associated memory;

application component means for providing data management functions;

service component means for providing data transformation and monitoring functions;

presentation component means for providing data presentation functions;

means for operatively coupling each of said application component means, said service component means to said computer processing means, and

control component means for controlling and said presentation component means to process said transaction in accordance with a business model.

42. A system according to claim 41 wherein said application component means is adapted for providing a plurality of data management functions with respect to a single database.

43. A system according to claim 41 wherein said service component means is adapted for invoking functions provided by said application component means.

44. A system for processing a transaction comprising:

a computer processing system and associated memory;

a control component adapted for controlling said computer processing system to process said transaction as a function of a predefined sequence of commands associated with a business model;

each of said commands being associated with a function utilized to computer processing system to process said transaction, and each of said commands being selected from the group consisting of

application commands for invoking functions associated with data management;

service commands for invoking functions associated with data transformation and monitoring; and

presentation commands for invoking functions associated with presenting data to participants of the transaction.

45. A method for processing a transaction, the steps of the method comprising:

providing a computer processing system and associated memory;

providing a plurality of independent component modules operatively coupled to the computer processing system, wherein each component module is selected from the group of functions consisting of

data management functions utilizing an application component module;

data transformation and monitoring functions utilizing a service component module;

data presentation functions utilizing a presentation component module; and

control functions utilizing at least one application component module, service component module or presentation component module to process said transaction.

46. The method of claim 45 further comprising the steps of:

invoking a plurality of said component modules as a function of a predefined sequence of functions provided by a business action.

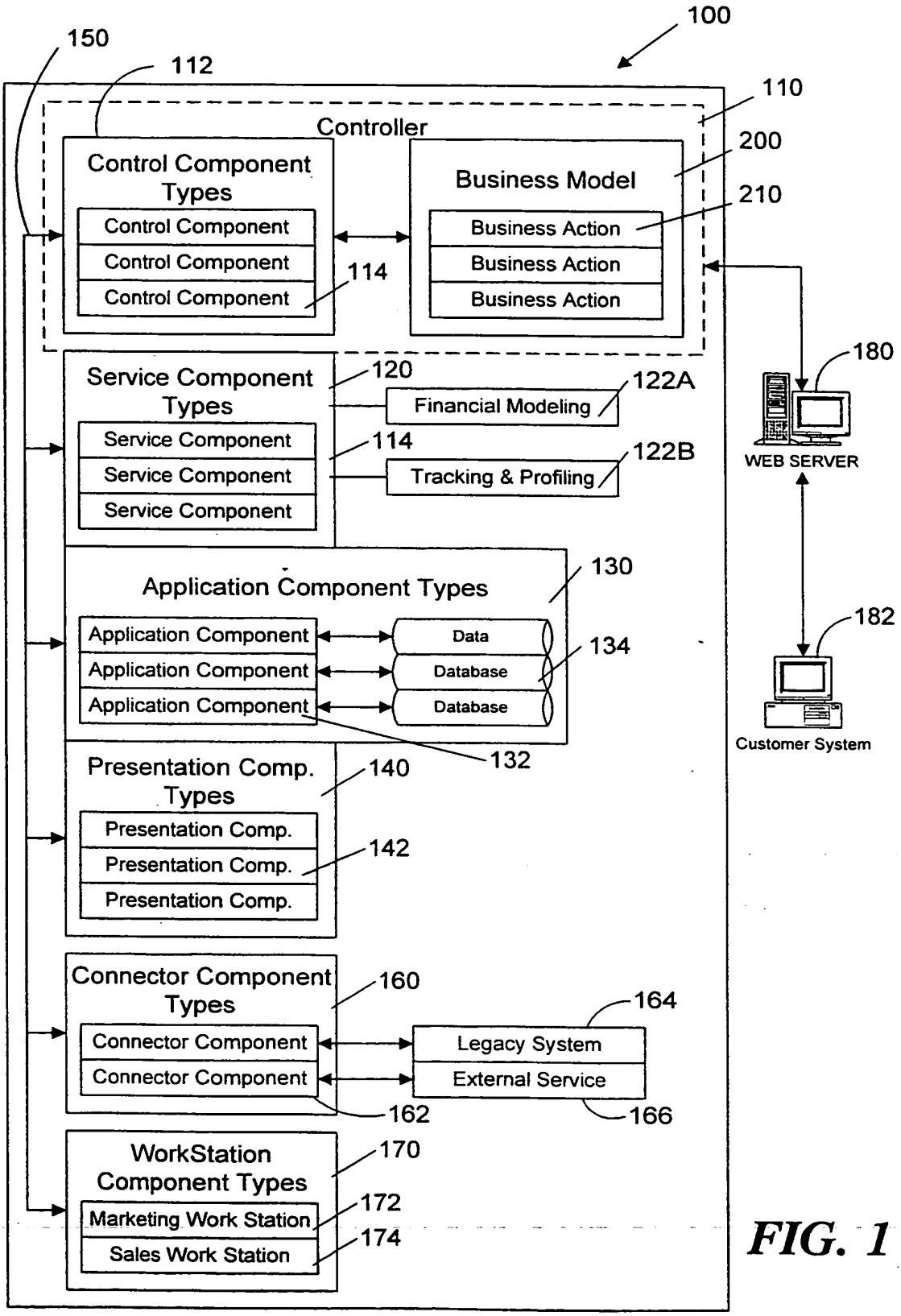
100
150
112
110
Controller
Control Component Types
Control Component
Control Component
Control Component
114
Business Model
200
210
Business Action
Business Action
Business Action
180
WEB SERVER
Service Component Types
120
Financial Modeling
122A
Service Component
Service Component
Service Component
114
Tracking & Profiling
122B
Application Component Types
130
Application Component
Application Component
Application Component
Data
Database
Database
134
132
182
Customer System
Presentation Comp. Types
140
Presentation Comp.
Presentation Comp.
Presentation Comp.
142
Connector Component Types
160
164
Connector Component
Connector Component
Legacy System
External Service
162
166
WorkStation Component Types
170
Marketing Work Station
Sales Work Station
172
174

FIG. 1

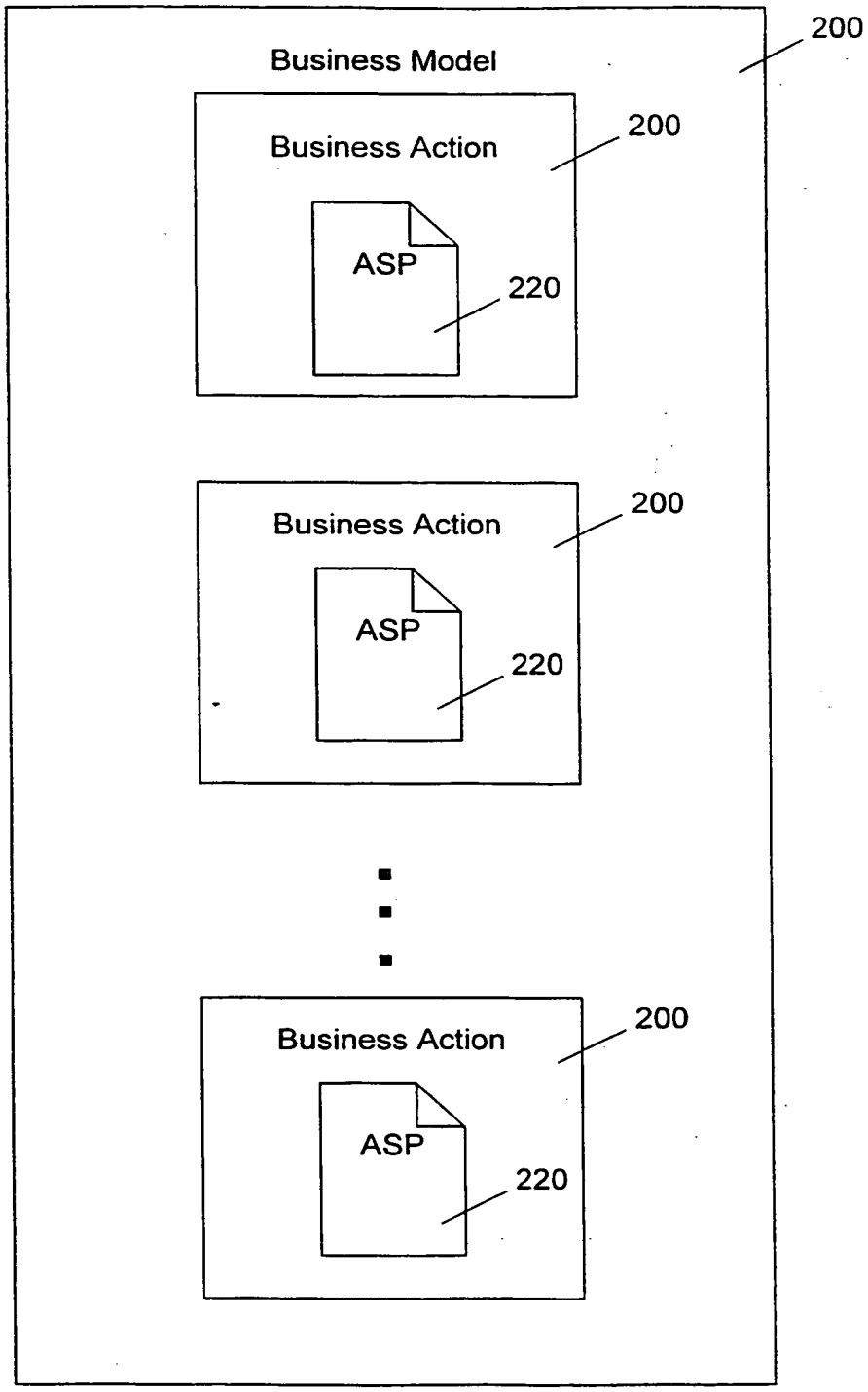

FIG. 2

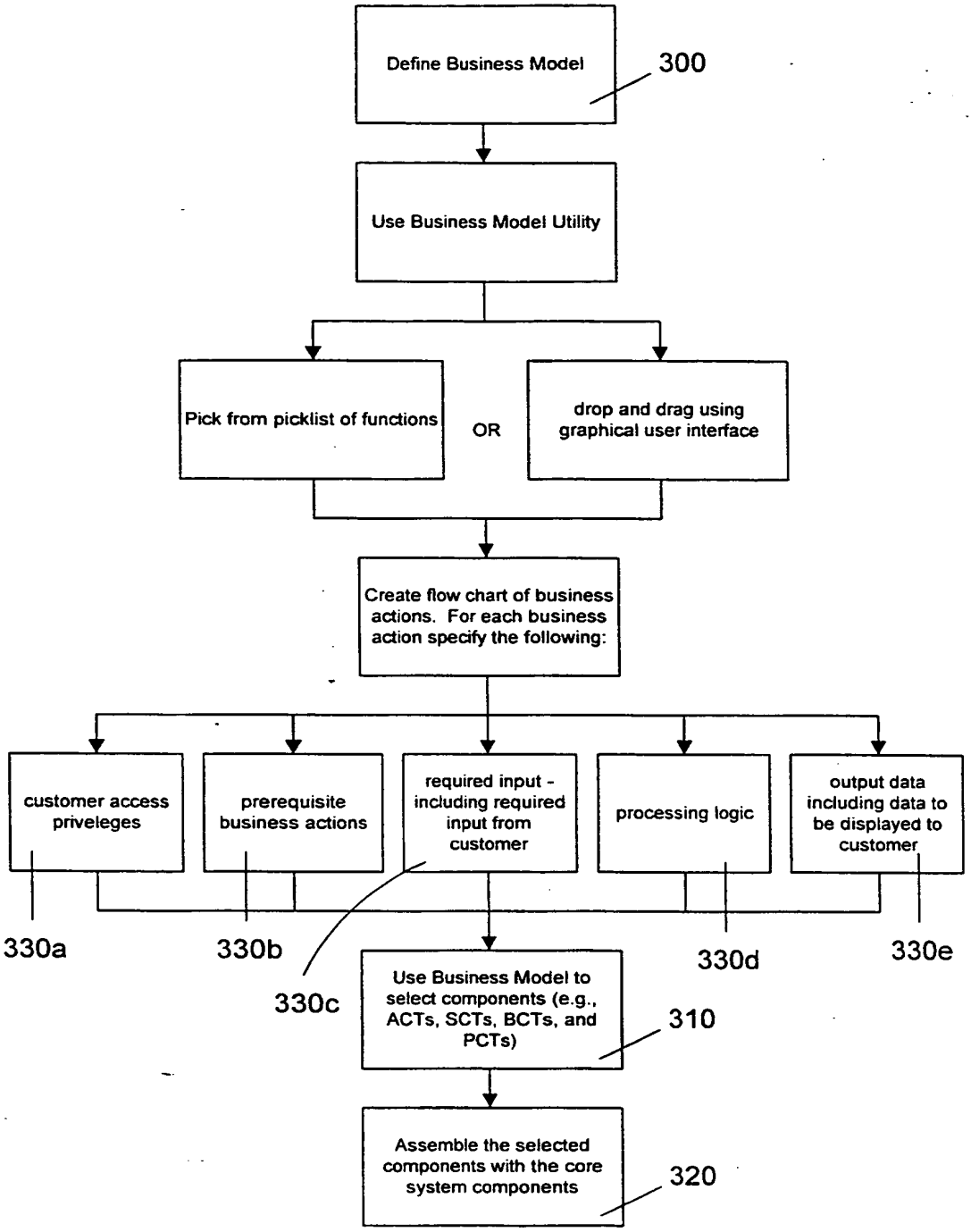

FIG. 3

Flow chart for constructing an e-commerce system

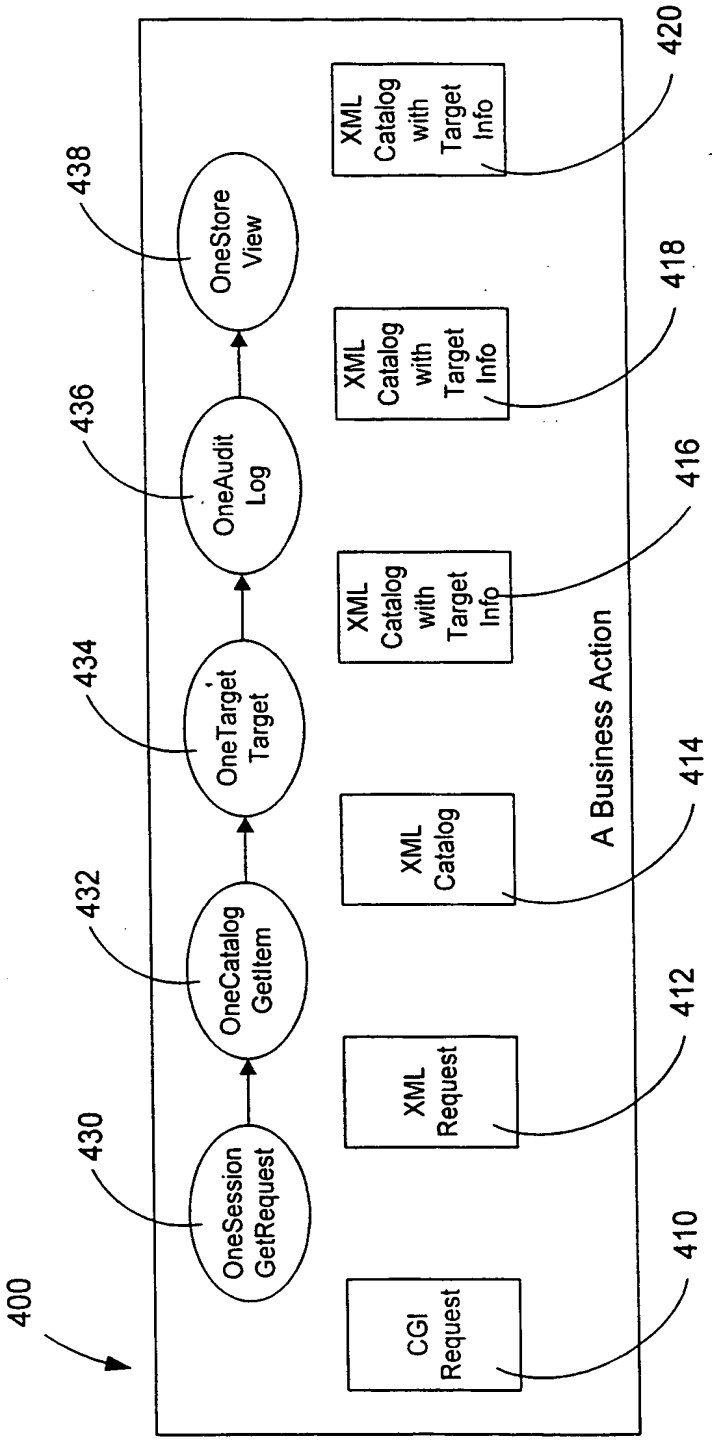

FIG. 4

```
<ONEITEM TYPE="OneSession" ID="0b6609fd730d99cfa1067445979c438d" VERSION="1"
SITE="Sportswarehouse" >
        . . .
        <ONEITEM TYPE="OneCustomer" ID="vmontan" VERSION="1" SITE="Sportswarehouse" />
        <ONEITEM TYPE="OneOrder" ID="9873635" VERSION="1" SITE="Sportswarehouse" />
        . . .
        <VARIABLE1>A Session specific variable value added by a service</VARIABLE1>
        <VARIABLE2>Another Session specific variable value</VARIABLE2>
        <VARIABLE3>A third value with <CHILD>children</CHILD></VARIABLE3>
        . . .
</ONEITEM>
```

The OneSession data item holds information generated by and used by the service components.

FIG. 5

```
<ONEITEM TYPE="OneSession" ID="0b6609fd730d99cfa1067445979c438d" VERSION="1" SITE="Sportswarehouse" >
    ...
    <REQUEST>
        <VARIABLES>
            <ACTION>ViewCatalog</ACTION>
            <ID>203</ID>
        </VARIABLES>
        <SERVER_VARIABLES>
            <APPL_PHYSICAL_PATH>C:\Inetpub\wwwroot\</APPL_PHYSICAL_PATH>
            <LOGON_USER/>
            <REMOTE_ADDR>204.176.12.67</REMOTE_ADDR>
            <HTTP_ACCEPT>*/*</HTTP_ACCEPT>
            <HTTP_ACCEPT_LANGUAGE>en-us</HTTP_ACCEPT_LANGUAGE>
            <URL>ViewCatalog.asp</URL>
            <HTTP_USER_AGENT>Mozilla/4.0 (compatible; MSIE 5.0b2; Windows NT)</HTTP_USER_AGENT>
            <HTTP_REFFERER/>
        </SERVER_VARIABLES>
        <COOKIES>
            <SITESERVER>ID=0b6609fd730d99cfa1067445979c438d</SITESERVER>
        </COOKIES>
        <CLIENT_CERTIFICATES/>
    </REQUEST>
    ...
</ONEITEM>
```

The OneSession data item holds user request information to be used by the service components

FIG. 6

```
<ONEITEM TYPE="OneSession" ID="0b6609fd730d99cfa1067445979c438d" VERSION="1" SITE="Sportswarehouse" >
    ...
    <ONEITEM TYPE="OneAudit" >
        <SESSION_ID>0b6609fd730d99cfa1067445979c438d</SESSION_ID>
        <USER_ID>vmontan</USER_ID>
        <TYPE>OneOrder</TYPE>
        <ACTION>AddToOrder</ACTION>
        <INFO>optional additional service info</INFO>
        <STATUS>Success</STATUS>
        <END_TIME>11/12/1998 11:57:52</END_TIME>
        <ACT>
            <TYPE>OneOrder</TYPE>
            <ACTION>GetItem</ACTION>
            <INFO>91234</INFO>
            <END_TIME>11/12/1998 11:57:40</END_TIME>
            <STATUS>Success</STATUS>
        </ACT>
        <ACT>
            <TYPE>OneOrder</TYPE>
            <ACTION>SetItem</ACTION>
            <INFO>91234</INFO>
            <END_TIME>11/12/1998 11:57:48</END_TIME>
            <STATUS>Success</STATUS>
        </ACT>
    </ONEITEM>
    ...
</ONEITEM>
```

The OneSession data item holds an audit trail of all actions that were performed in a session.

FIG. 7

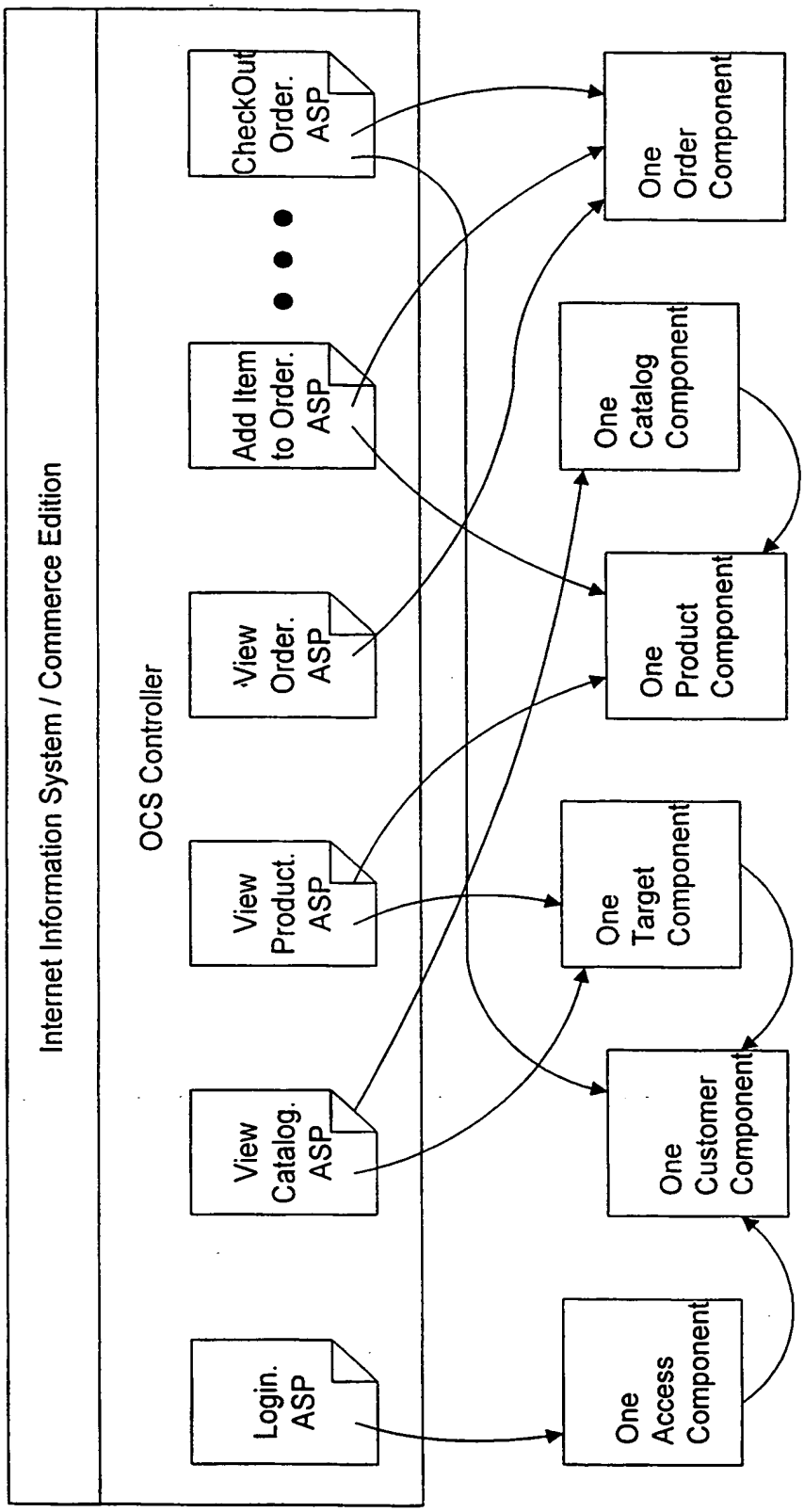
Internet Information System / Commerce Edition
OCS Controller
Login. ASP
View Catalog. ASP
View Product. ASP
View Order. ASP
Add Item to Order. ASP
CheckOut Order. ASP
One Access Component
One Customer Component
One Target Component
One Product Component
One Catalog Component
One Order Component

FIG. 8

```
<%@ Language=JavaScript %>
<!--#include file="ICS/functions.inc" -->
<%
        // Begin a request by converting request to XML
        sXML = ICSCallService("OneSession", "BeginRequest", "");

        // Get the contents of the catalog that was requested
        sXML = ICSCallService("OneCatalog", "GetCatalog", sXML);

        // Remove items which the user does not have access to
        xXML = ICSCallService("OneAccess", "AccessData", sXML);

        // Update targetting information and add in targetted products
        sXML = ICSCallService("OneTarget", "TargetProducts", sXML);

        // Get the appropriate view for the user
        sHTML = ICSCallService("OneSite", "ViewResults", sXML);
        Response.Write(sHTML);

        // Log the request and outcome of services to perform the request
        sXML = ICSCallService("OneAudit", "Log", sXML);

        // Save Session information
        sXML = ICSCallService("OneSession", "EndRequest", sXML);%>
```

An ASP file which handles the request to view a targeted catalog.

FIG. 9

```
<ONEITEM ID="1" TYPE="OneProduct" VERSION="1" SITE="SampleSite">
        <NAME>My Hard Good</NAME>
        <DESCRIPTION>This is the first hard good that I have for sale in my store.</
DESCRIPTION>
        <IMAGE_URL> /images/hardgood1.gif</IMAGE_URL>
        <PRICE>19.99</PRICE>
        <CREATOR>lbailey</CREATOR>
        <STATUS>1</STATUS>
        <VISIBLE>1</VISIBLE>
        <UPDATED_BY>lbailey</UPDATED_BY>
        <CREATION_DATE>10/12/69</CREATION_DATE>
        <UPDATE_DATE>10/12/69</UPDATE_DATE>
        <WEIGHT>1.4 lbs</WEIGHT>
        <HEIGHT>1 foot</HEIGHT>
        <WIDTH>1 foot</WIDTH>
        <DEPTH>1 foot</DEPTH>
        <THUMBNAIL>/images/hg1_thumbnail.gif</THUMBNAIL>
        <SKU>HG1IDENT</SKU>
        <EFFECTIVE_DATE>10/25/98</EFFECTIVE_DATE>
        <EXPIRATION_DATE>1/1/2000</EXPIRATION_DATE>
        <PRODUCT_STATUS>Available</PRODUCT_STATUS>
        <FEATURE ID="1">
                <CATEGORY>Size</CATEGORY>
                <VALUE ID="23">S</VALUE>
                <VALUE ID="24">M</VALUE>
                <VALUE ID="26">L</VALUE>
                <VALUE ID="27">XL</VALUE>
        </FEATURE>
        <FEATURE ID="2">
                <CATEGORY>Color</CATEGORY>
                <VALUE ID="29">Green</VALUE>
                <VALUE ID="32">Red</VALUE>
                <VALUE ID="34">Blue</VALUE>
        </FEATURE>
        <MANUFACTURERS>
                <COMPANY ID="12">
                        <NAME>ACME Explosives</NAME>
                        <BRANDNAME>ACME</BRANDNAME>
                        <DESCRIPTION>Makers of things that go Boom.</DESCRIPTION>
                        <COMPANY_IMAGE>/images/acme.gif</COMPANY_IMAGE>
                        <COMPANY_URL>http://www.acme.com/</COMPANY_URL>
                </COMPANY>
        </MANUFACTURERS>
        <VENDORS>
                <COMPANY ID="13">
                        <NAME>ACME Retail</NAME>
                        <BRANDNAME>ACME</BRANDNAME>
                        <DESCRIPTION>Distributor of things that go Boom.</DESCRIPTION>
                        <COMPANY_IMAGE>/images/acme.gif</COMPANY_IMAGE>
                        <COMPANY_URL>http://www.acme.com/</COMPANY_URL>
                <COMPANY>
        </VENDORS>
</ONEITEM>
```

Example of a Product Application Component Data Item

FIG. 10

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

REVISED VERSION

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 August 2000 (10.08.2000)

PCT

(10) International Publication Number
WO 00/46723 A2

(51) International Patent Classification[7]: G06F 17/60

(21) International Application Number: PCT/US00/02933

(22) International Filing Date: 3 February 2000 (03.02.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/118,493	3 February 1999 (03.02.1999)	US
09/496,530	2 February 2000 (02.02.2000)	US

(71) Applicant: **ONESOFT CORPORATION** [US/US]; Suite 250, 7010 Little River Turnpike, Annandale, VA 22003-9998 (US).

(72) Inventors: **MACINTYRE, James, W.**; 4613 Hillbrook Drive, Annandale, VA 22003 (US). **KOBEISSI, Said**; 5810 Wood Poppy Court, Burke, VA 22015 (US). **PARKER, Eric**; 245 Summer Street, Somerville, MA 02143 (US). **MONTAN, Vasile**; 6008 Lincolna Road, Alexandria, VA 22312 (US). **BAILEY, Robert, L.**; 8613 Running Fox Court, Fairfax Station, VA 22039 (US).

(74) Agent: **MIRABITO, A., Jason**; Mintz Levin Cohn Ferris Glovsky and Popeo PC, One Financial Center, Boston, MA 02111 (US).

(81) Designated States *(national)*: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with declaration under Article 17(2)(a); without abstract; title not checked by the International Searching Authority*

(48) Date of publication of this revised version: 27 June 2002

(15) Information about Correction:
see PCT Gazette No. 26/2002 of 27 June 2002, Section II

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 00/46723 A2

(54) Title: MODULAR SYSTEM AND METHOD FOR PROCESSING TRANSACTIONS

(57) Abstract:

PATENT COOPERATION TREATY

PCT

DECLARATION OF NON-ESTABLISHMENT OF INTERNATIONAL SEARCH REPORT

(PCT Article 17(2)(a), Rules 13ter.1(c) and Rule 39)

Applicant's or agent's file reference	IMPORTANT DECLARATION	Date of mailing *(day/month/year)*
16587-030		26/03/2002
International application No.	International filing date *(day/month/year)*	(Earliest) Priority date *(day/month/year)*
PCT/US 00/ 02933	03/02/2000	03/02/1999
International Patent Classification (IPC) or both national classification and IPC	G06F17/60	
Applicant		
ONESOFT CORPORATION		

This International Searching Authority hereby declares, according to Article 17(2)(a), that **no international search report will be stablished** on the international application for the reasons indicated below

1. [X] The subject matter of the international application relates to:
   - a. [ ] scientific theories.
   - b. [ ] mathematical theories
   - c. [ ] plant varieties.
   - d. [ ] animal varieties.
   - e. [ ] essentially biological processes for the production of plants and animals, other than microbiological processes and the products of such processes.
   - f. [X] schemes, rules or methods of doing business.
   - g. [ ] schemes, rules or methods of performing purely mental acts.
   - h. [ ] schemes, rules or methods of playing games.
   - i. [ ] methods for treatment of the human body by surgery or therapy.
   - j. [ ] methods for treatment of the animal body by surgery or therapy.
   - k. [ ] diagnostic methods practised on the human or animal body.
   - l. [ ] mere presentations of information.
   - m. [ ] computer programs for which this International Searching Authority is not equipped to search prior art.

2. [ ] The failure of the following parts of the international application to comply with prescribed requirements prevents a meaningful search from being carried out:
   - [ ] the description
   - [ ] the claims
   - [ ] the drawings

3. [ ] The failure of the nucleotide and/or amino acid sequence listing to comply with the standard provided for in Annex C of the Administrative Instructions prevents a meaningful search from being carried out:
   - [ ] the written form has not been furnished or does not comply with the standard.
   - [ ] the computer readable form has not been furnished or does not comply with the standard.

4. Further comments:

Name and mailing address of the International Searching Authority
European Patent Office, P.B. 5818 Patentlaan 2
NL-2280 HV Rijswijk
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,
Fax: (+31-70) 340-3016

Authorized officer

Georgios Chatzaras

FURTHER INFORMATION CONTINUED FROM PCT/ISA/ 203

The claims relate to subject matter for which no search is required according to Rule 39 PCT. Given that the claims are formulated in terms of such subject matter or merely specify commonplace features relating to its technological implementation, the search examiner could not establish any technical problem which might potentially have required an inventive step to overcome. Hence it was not possible to carry out a meaningful search into the state of the art (Art. 17(2)(a)(i) and (ii) PCT; see Guidelines Part B Chapter VIII, 1-6).

The applicant's attention is drawn to the fact that claims relating to inventions in respect of which no international search report has been established need not be the subject of an international preliminary examination (Rule 66.1(e) PCT). The applicant is advised that the EPO policy when acting as an International Preliminary Examining Authority is normally not to carry out a preliminary examination on matter which has not been searched. This is the case irrespective of whether or not the claims are amended following receipt of the search report or during any Chapter II procedure. If the application proceeds into the regional phase before the EPO, the applicant is reminded that a search may be carried out during examination before the EPO (see EPO Guideline C-VI, 8.5), should the problems which led to the Article 17(2) declaration be overcome.